GSWC 2013

Proceedings of

The Eighth Annual Graduate Student Workshop on Computing

October 4th, 2013

Santa Barbara, California

Department of Computer Science

University of California, Santa Barbara

http://www.cs.ucsb.edu

Organized By

Gianluca Stringhini, Chair

Mariya Zheleva, Vice-Chair

Saiph Savage, Industry Liaison

Nazli Dereli, Financial Coordinator

Kyle Klein, Proceedings Coordinator

Sean Maloney, Website Coordinator

Adam Doupé, General Committee

Alexander Pucher, General Committee

Ali Zand, General Committee

Dhilung Kirat, General Committee

Faisal Nawab, General Committee

Morgan Vigil, General Committee

Paul Schmitt, General Committee

Qingyun Li, General Committee

Stephen Gauglitz, General Committee

Sourav Medya, General Committee

Vaibhav Arora, General Committee

Yanick Fratantonio, General Committee

Platinum Supporters

Bronze Supporters

Thanks to

AchieveMint

Keynote Speaker

Bio: Mike Brzozowski is a user experience researcher focusing on Google's social products. His research interests include social computing and online communities. Previously, he was a research scientist in the Social Computing Lab at HP Labs, and specialized in human-computer interaction in the Stanford Computer Science Department.

Title: Beyond Big Data: Understanding Users at Scale

Abstract:
Online social media offers an unprecedented laboratory to explore social interactions at scale. Yet it's hard to anticipate all the effects of a system with so many actors, and the systems we build are often re-appropriated by end users. To better understand how these systems are used, we employ mixed methods research to combine small-scale observations with large-scale data analysis. I'll provide a few examples of how we integrate quantitative and qualitative methods to better understand our users.

Industry Panelists

Luca Foschini is a co-founder AchieveMint, a platform that uses data analytics on people's activities to incentivize healthy behavior. His focus at AchieveMint is on data science and mobile integration. Luca earned a Ph.D. in Computer Science from UC Santa Barbara where he developed efficient algorithms for routing in road networks under heavy traffic conditions, which earned him the Dean's Fellowship and internships at Google Research and the ETH Zurich. He also holds a Master in Engineering from the University of Pisa, and is an alumnus of the Sant'Anna School of Advanced Studies. In a previous life, Luca accumulated 5 years of industry experience at Ask.com, Google, and the CERN, and was a coach of the Italian national team participating in the International Olympiad in Informatics (IOI).

Siddharth Jayaraman is a Software Engineer with Qualcomm Research working on Context-Aware computing. Prior to this, he was part of the protocol stack team for the broadcast technology (MediaFLO) division of Qualcomm. He got his masters in CS from SUNY at Buffalo with a major in Artificial Intelligence.

Ahmed Metwally holds a M.S and a Ph.D. from UC Santa Barbara, and a B.S. from Alexandria University, all in Computer Science. His main interests are designing scalable data analysis algorithms for emerging architectures, traffic stream analysis, and clustering, with an application to abuse detection. Ahmed has been with Google for five years working on traffic anomaly detection.

Dr. Klaus Schauser is a Founder and Chief Strategist of AppFolio, a fast growing Software-as-a-Service Startup in Santa Barbara that raised $30M of VC funding. AppFolio creates complete, easy-to-use, business solutions for multiple vertical markets. Products include the SecureDocs secure virtual data rooms for financial events and corporate archiving, the MyCase web-based legal practice management software, and the AppFolio Property Manager web-based property management software. Klaus brings his enthusiasm, leadership and experience to the team. When he's not sharing his vision, validating new product offers or setting the strategy you can find him outdoors hiking or stand-up paddleboarding, enjoying the beauty of Santa Barbara.

Klaus was a Founder and CTO of Expertcity/CitrixOnline from 1999 through 2006 and was the visionary behind GoToMyPC, GoToAssist, and GoToMeeting. He led the teams responsible for building the products and their secure, reliable Software-as-a-Service (SaaS) infrastructure. As a Professor of Computer Science at the University of California, Santa Barbara, Klaus is a widely published research scientist with extensive experience developing scalable, highly parallel computing environments. Klaus holds a Ph.D. from the University of California, Berkeley and has received numerous academic awards.

Table of Contents

Session 1: Hardware (Moderated by Gianluca Stringhini)

- SurfNoC: A Low Latency and Provably Non-Interfering Approach to Secure Networks-On-Chip 1
  Hassan M. G. Wassel, Frederic T. Chong, Timothy Sherwood
- Visualizing Information Flow at the Circuit Level 3
  Bita Mazloom, Jason Oberg, Ryan Kastner, Tim Sherwood
- Memristors for Neural Branch Prediction: A Case Study in Strict Latency and Write Endurance Challenges 5
  Hebatallah Saadeldeen, Diana Franklin, Timothy Sherwood, Frederic T. Chong, Dmitri Strukov

Session 2: Theory (Moderated by Kyle Klein)

- Fast Clustering Methods for Genetic Mapping 7
  Veronika Strnadova, John Gilbert, Aydin Buluc, Leonid Oliker, Joseph Gonzalez, Stefanie Jegelka
- A Simple, Efficient Preconditioner for Graph Laplacians 9
  Erik G. Boman, Kevin Deweese
- On the Most Likely Convex Hull of Uncertain Points in the Plane 11
  Hakan Yıldız

Session 3: Connecting People (Moderated by Nazli Dereli)

- Android at Bandon Bay: Low-Cost Environmental Monitoring and Event Detection Using Smartphones 13
  Michael Nekrasov, Sirilak Chumkiew, Peter Shin
- You are How You Click: Sybil Detection Using Clickstream Models 15
  Gang Wang, Tristan Konolige, Christo Wilson, Xiao Wang, Haitao Zheng, Ben Y. Zhao
- Augmented Reality-based Remote Collaboration 17
  Steffen Gauglitz, Matthew Turk, Tobias Hollerer

Session 4: Programming Languages (Moderated by Yanick Fratantonio)

- Hairball: Lint-inspired Static Analysis of Scratch Projects 19
  Bryce Boe, Phillip Conrad, Diana Franklin
- The Design of notJS: an Intermediate Representation of JavaScript for Abstract Interpretation 21
  Vineeth Kashyap, Kyle Dewey, Ethan A. Kuefner, Ben Hardekopf

- Dynamic Machine-Code Generation for Quantum Rotations 23
  Daniel Kudrow, Kenneth Bier, Zhaoxia Deng, Diana Franklin, Frederic T. Chong

Posters

- Detecting Twitter Compromises 25
  Gianluca Stringhini, Manuel Egele
- Creepic: I know who you did last summer. 27
  Yan Shoshitaishvili, Christopher Kruegel, Giovanni Vigna
- Kwiizya: Local Cellular Network Services in Remote Areas 29
  Mariya Zheleva, Arghyadip Paul, David L. Johnson, Elizabeth Belding
- Writing Groups in Computer Science Research Labs 31
  Adam Doupé and Janet L. Kayfetz
- Submit: a mobile communication control platform 33
  Morgan Vigil, Waylon Brunette, Gaetano Boriello
- Critic: Designing a Sound Static Analyzer for Chisel 35
  Ethan A. Kuefner, Timothy Sherwood, Ben Hardekopf, John Sarracino
- On the Self-similarity of Social Network Dynamics 37
  Qingyun Liu, Xiaohan Zhao, Stephanie Smith, Ben Y. Zhao,
  Haitao Zheng, Walter Willinger
- Say My Name, Say My Name: User Mentioning on Facebook 39
  Saiph Savage, Andres Monroy-Hernandez, Leif Singer, Tobias Hollerer

SurfNoC: A Low Latency and Provably Non-Interfering Approach to Secure Networks-On-Chip

Hassan M. G. Wassel Frederic T. Chong Timothy Sherwood
Department of Computer Science, UC Santa Barbara
{hwassel, chong, sherwood}@cs.ucsb.edu

I. Introduction

Programmers are increasingly asked to manage a complex collection of computing elements including a variety of cores, accelerators, and special purpose functions. While these many-core architectures can be a boon for common case performance and power-efficiency, when an application demands a high degree of reliability or security the advantages becomes a little less clear. On one hand, the ability to spatially separate computations means that critical operations can be physically isolated from malicious or untrustworthy components. There are many advantages to providing physical separation which have been well explored in the literature. On the other hand, real systems are likely to use different subsets of cores and accelerators based on the needs of the application and thus will require a shared communication network. When a general purpose interconnect is used, analyzing all the ways in which an attacker might influence the system becomes far more complicated. The problem is hard enough if we restrict ourselves to considering only average case performance or packet ordering, but the difficulty of the problem increases even further if we attempt to *prevent even cycle-level variations*.

In high assurance systems it is common practice to break the system into a set of domains which are to be kept separate. These domains should have *no-effect* on one another. For example, the Mars Curiosity rover software runs on a RAD750 processor, a single-core radiation-hardened version of the Power architecture with a special purpose separation kernel. The kernel partitions the tasks such as guidance, navigation and the various science packages from one another to help prevent cascading failures. Future space missions are looking to use multicore systems which adds another layer of communication, but there are serious concerns about the introduction of opportunities for interference between system components.

The problem is that typical networks-on-chip have many internal resources that are shared between packets from different domains which we would otherwise wish to keep separate. These resources include the buffers holding the packets, the crossbar switches, and the individual ports and channels. Such resource contention introduces "interference" between these different domains which can create a performance impact on some flows, pose a security threat by creating an opportunity for timing channels [2], and generally complicates the final verification and certification process of the system because *all* of the ways in which that interaction might occur must be accounted for. Non-interference means that injection of packets from one domain cannot affect the timing of delivery of packets from other domains.

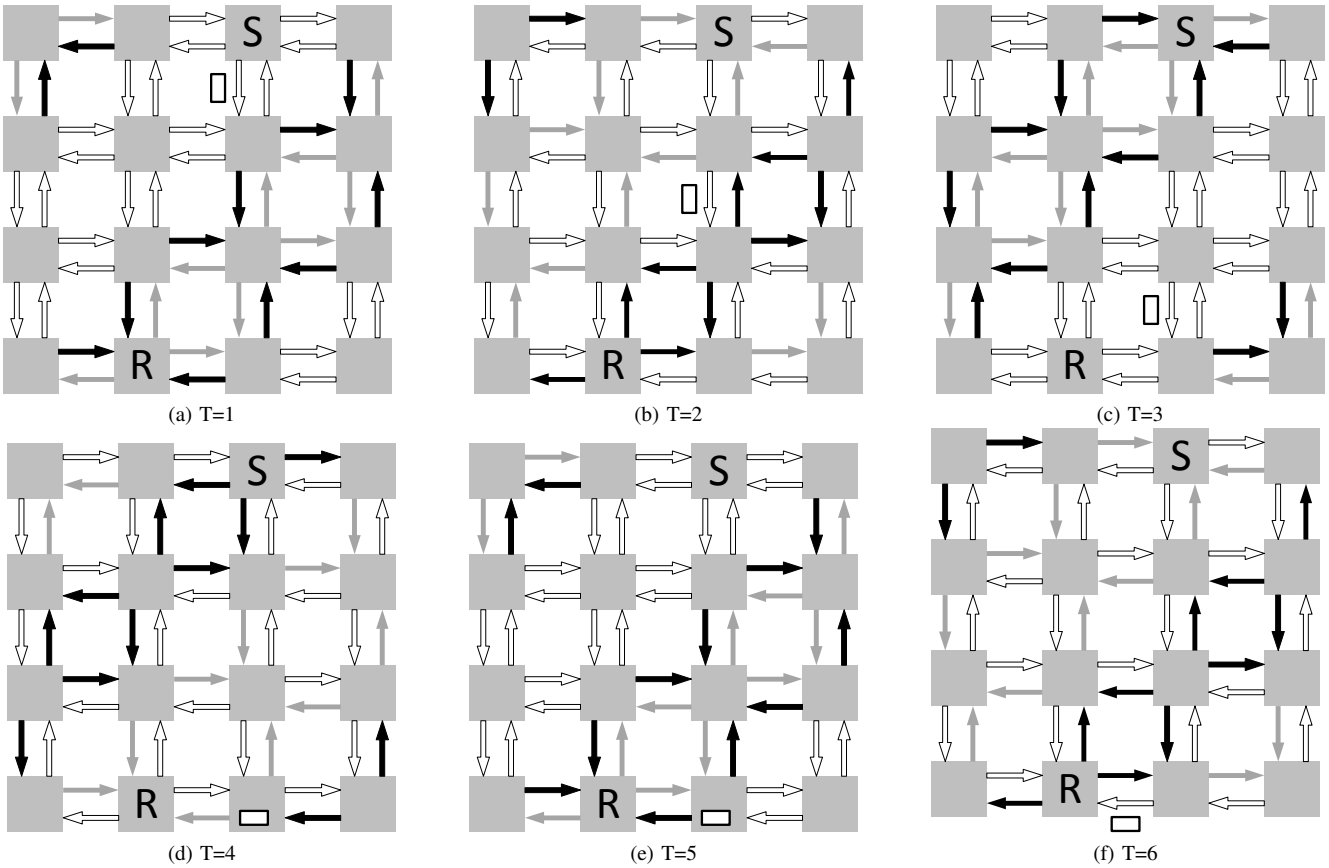
These concerns are similar to, but distinct from, the problem of providing quality-of-service guarantees. While QoS can minimize the performance impact of sharing between domains by providing a minimum guaranteed level of service for each domain (or class), as shown by Wang and Suh, quality of service techniques will still allow timing variations and thus do not truly support non-interference [2]. The only way to be certain that the domains are non-interfering is to statically schedule the domains on the network over time. However, a straightforward application of time multiplexing leads to significant increases in latencies as each link in the network is now time multiplexed between many different domains.

II. SurfNoC

The most basic routing algorithm in meshes and tori is dimension-ordered routing. That is, a packet walks through a dimension until it cannot move further without going farther from the destination and then transfers to an other dimension. Thus, routing is linear in each dimension which provides an opportunity to reduce wait time between hops. This way packets will only have to wait when they enter (exit) the network from (to) the injection (ejection) channel and when they change dimensions. We will describe this idea in details in the rest of this section.

The straightforward way to support time-division multiplexing is to operate the whole network in time slices that are divided between application domains. That is a packet waits at each hop until the network is forwarding packets from that its domain. This approach leads to a zero-load latency L_0 that is proportional to the number of application domains D, pipeline depth P, and the number of hops H. This solution might work efficiently for a small number of domains such as 2 to 4 domains but in high assurance applications as many as tens of domains can be found [1].

We propose SurfNoC scheduling in which different routers (and in fact different ports of the same router) can forward packets from different domains at the same cycle. In this schedule, a packet waits until it can be forwarded in one dimension (i.e. its output channel is forwarding packets from its domain at this cycle) and then does not experience any wait at any downstream router in this dimension (assuming there is no contention from packets from the same domain) in a way similar to the schedule developed in the half-mesh example. After finishing the first dimension, the packet may experience another wait until it can be forwarded on the next dimension. We call this schedule Surf scheduling because a packet is like



(a) T=1 (b) T=2 (c) T=3

(d) T=4 (e) T=5 (f) T=6

Fig. 1: Surf scheduling in 16-node 2D mesh with three application domains (denoted by white, grey, and black) assuming a single-cycle routers for illustration purpose. The schedule runs as white, white, grey, and black and repeats, giving the white domain half the bandwidth. A packet (the white box under the node S) belongs to the white domain is sent from the node marked by S to the node marked by R. The figure contains six consecutive cycles. At $T = 1$, the packet is forwarded on the S port in the y-dimension (which is scheduled to forward white packets). It keeps moving in the y-dimension until $T = 3$ when it needs to move in the x-dimension on the W port. The packet waits 2 cycles (T=4 and T= 5) until it is the white domain's turn on the W port and finally it is forwarded to its destination on $T = 6$. Another wait may happen again in the destination router (R) to forward the packet on the ejection port waiting for the white domain's turn.

a surfer who waits to "ride" a wave until some location and then waits to "ride" another wave.

Figure 1 illustrates an example of 16-node 2D mesh schedule of 3 domains (colored white, grey, black). There are two waves south-east (SE) and north-west (NW) running in the mesh. Each channel propagates packets according to the following schedule (white, white, gray, and black) and repeats. It is worth noting that using such a schedule results in half of the bandwidth being allocated to the white domain, whereas the black and grey domains guarantee only a quarter of the bandwidth for each of them. This illustrates the benefit of our schedule in statically allocating non-uniform bandwidth to domains.

III. Conclusions

In this paper, we introduced SurfNoC, an on-chip network that significantly reduces the latency incurred by temporal partitioning. By carefully scheduling the network into waves that flow across the interconnect, data with different labels carried by these wave are strictly non-interfering while avoiding the significant overheads associated with cycle-by-cycle time multiplexing. We evaluate the scheme for different number of labels, network sizes, queue sizes. We show that performance overhead over the non-protected scheme is reduced from 39.34(56) cycles using the base line time-division multiplexing to 35(48) using the Surf Schedule, a latency improvement of 10%(14%) for a 64-node network under uniform random traffic pattern 2(8) domains. For more details about the technique, verification of security property and evaluation of performance, please refer to our paper [3]

References

[1] J. Rushby. Partitioning for Avionics Architectures: Requirements, Mechanisms, and Assurance. NASA Contractor Report CR-1999-209347, NASA Langley Research Center, June 1999. Also to be issued by the FAA.

[2] Y. Wang and G. Suh. Efficient timing channel protection for on-chip networks. In *Networks on Chip (NoCS), 2012 Sixth IEEE/ACM International Symposium on*, pages 142 –151, may 2012.

[3] H. M. G. Wassel, Y. Gao, J. K. Oberg, T. Huffmire, R. Kastner, F. T. Chong, and T. Sherwood. Surfnoc: a low latency and provably non-interfering approach to secure networks-on-chip. In *Proceedings of the 40th Annual International Symposium on Computer Architecture*, ISCA '13, pages 583–594, New York, NY, USA, 2013. ACM.

Visualizing Information Flow at the Circuit Level

Bita Mazloom*, Jason Oberg†, Ryan Kastner† and Tim Sherwood*
*Computer Science, UC Santa Barbara, {betamaz, sherwood}@cs.ucsb.edu
†Computer Science and Engineering, UC San Diego, {joberg, kastner}@cs.ucsd.edu

I. Abstract

***Abstract*—The enforcement of information flow policies in modern computer systems is one of the most important aspects of modern computer security, yet it is one of the hardest to get correct in implementation. Current tools only provide a primitive waveform representation that the designers must tediously compare and contrast to verify the correctness of their design. We argue that although the data might be primitive, interacting devices result in complex behaviors that require a more effective scalable representation that enable designers to see the overall picture of the system behavior. We propose moving beyond the waveform representation to a model that takes into account the relationship between different parts of the circuit being designed.**

II. Introduction

Today's machines are composed of multiple layers where each layer serves as a vital organ to the successful survival of the system at large. To ensure the system provides the expected operation in spite of the environment in which it resides, it is crucial to understand how the proposed system behaves given different conditions. In the case of secure embedded systems, the key factor is the trustworthiness of data. Anomalous events at the lowest hardware layer have cascading side effects throughout the system, which could potentially result in performance degradation and security leaks.

The impact of information flow at the hardware layer is evident in Boeing's 787 model, where a shared physical network is used to manage both aircraft control data and passenger data. Information in the former is vital to the correct operation of the system and should remain secure and trusted. On the other hand, passenger data is assumed to not be trustworthy. If security of the data is not taken into account, a passenger receiving access to restricted information could make the system a potential risk.

Current work in embedded systems tries to address the problem of designing systems that are resilient to malicious users. However due to power and space limitations multiple components at the hardware layer share resources to improve performance, making incorporating security in the design an even more challenging problem. When hardware architects are designing a system that needs to provide a high level of assurance for the integrity of the data, each change requires an exhaustive test to determine that the implementation does not result in unforeseen security leaks. Therefore designers must manage the propagation of information at multiple regions to minimize against unforeseen channels of information flow.

As security cuts across many abstractions of hardware design, the challenge is how to visualize this property of the system across the entirety of the design. Although many design tools exist [1], [5], there is a disparity between the ease of designing and investigating the design. To understand how information propagates at the hardware layer, we propose delving into the relatively uncharted region of how to apply visualization techniques for diagnosing events at the gate level.

III. Related

Understanding whether untrusted data is effecting critical information in a microprocessor is similar to determining whether an attacker has compromised a network system. While there are various visualization tools for understanding network traffic [3], there is limited work in applying visualization techniques to aid in understanding of a processor's design. James et al. [4] uses high resolution photomicrographs to create a photo-realistic mapping of an old MOS 6502 processor. This visualization is useful for the study of historical microprocessor designs that have been fabricated. We take lessons learned from network analysis and information visualization [8] to provide designers with an at a glance view of circuit behavior while enabling them to pinpoint the root cause of a security leak.

IV. Approach

The design of a circuit is composed of combination of registers that store values and logic gates that perform the computation. Three of the most basic gates which perform bitwise operations are two input AND gates and OR gates and the one input NOT gate. A combinational logic circuit, or circuit as we use in this paper, refers to a combination of these logic gates that provide the same result given the same set of inputs. The correctness of a circuit's functionality is verified by testing for different independent input combinations. A circuit is said to be correct if given a set of inputs, the outputs have expected signal values. For a component in the GLIFT system, that means that untrusted information should have no bearing on the output trust state [7] of trusted data. When verifying that a component does not inadvertently propagate taint in the system, the designer must check that the outputs that need to be trusted are not tainted. So a representation of the circuit needs to enable the designer to determine how taint is propagated from input to output. To help the designer answer this question, the circuit view needs to show the connections between the gates. Showing the path from the primary input (PI) to the primary output (PO) that taint flows would enable the designer to more easily identify the areas of the design that incorrectly resulted in a security leak.

To enable designers to more easily relate to the representations, we drew insights from circuit drawing conventions [2], [6]. We laid out key features that were extracted from the circuit to more closely represent circuit schematics including orthogonal wires and reorienting the circuit view. After several

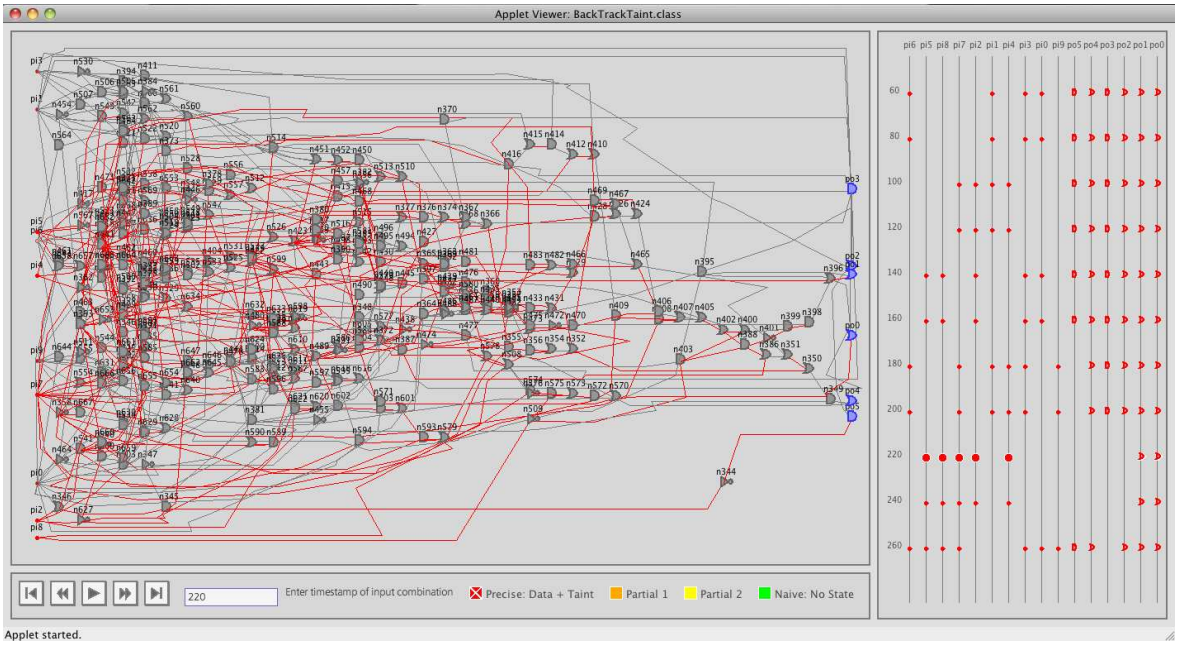


Fig. 1. The circuit view of the ALU2 (two input adder) combinational circuit with 9 primary inputs and 6 primary outputs. The color red indicates where taint has propagated to. The main view shows path of taint flow. The right panel gives summary of tainted input and output test combinations.

iterations we developed a representation that fit more with the model that the circuit designers expected with a minor compromise which we'll discuss shortly. We started by labeling the gate type with its symbol. This enabled the designer to easily determine the flow of information and enabled us to remove the edge direction as it is implied. Next, we distinguished the primary nets by their shape and color. Since the value of a PI is independent of a logic device, we used a simple dot shape. For a PO, as it is the output of a specific gate, we simply highlighted the corresponding gate. We transformed the edges to be closer in representation to the orthogonal routing used for wires in circuit diagrams. Additionally, we modified the gate placement to be drawn from left to right in a levelized order. Then, we shifted the PIs to reflect the expected location of inputs and pushed the POs on the rightmost boundary of the view space.

For a small circuit whose gates are easily countable, organizing the layout with the above modifications produced the desired effect. However, as the circuit became larger, the overlap of edges and gates increased to create a cluttered view. So, we eased the layout restriction to maintain an approximate level order of gates. For example, in Figure 1 the location of pi3, po3 and n564 where determined by their respective categories of primary input, primary output and internal gate. Also, we removed the necessity for edges to remain at orthogonal angles which provided a clearer view of the number of edges whose endpoints were at the primary nets. This was at the expense of moving away from the desired schematic view of the circuit.

When the designer is trying to understand how information flows from inputs to outputs, a single isolated test case will not give a complete picture. The designer checks the outputs against the inputs by testing different input combinations with specific values and trust settings. The circuit view needs to enable the designer to understand what happens across the test set. The main view only shows the flow of taint given a single input combination and does not indicate which test case is shown. Adding the additional snapshot view on the right enabled the designer to instantly get a high level overview of how tainted inputs effect the outputs across all test cases.

V. Conclusions

Based on work with domain experts, we present the first steps in the development of visualization tools for understanding the behavior of information flow security at the circuit layer. In the future, we plan to enable the designer to investigate data flow across multiple cycles allowing for the visualization of sequential cycles; and to more exhaustively use our tools for various embedded applications to determine if the same visual language can convey information flow properties to their architects.

References

[1] Altera Corporation. *Introduction to the Quartus II Software, v10*.

[2] T.-c. Chiueh. Heresy: a hybrid approach to automatic schematic generation. In *Proceedings of the conference on European design automation*, EURO-DAC '91, pages 419–423, Los Alamitos, CA, USA, 1991.

[3] J. R. Goodall. Introduction to visualization for computer security. pages 1–17, Apr. 2007.

[4] G. James, B. Silverman, and B. Silverman. Visualizing a classic cpu in action: the 6502. In *ACM SIGGRAPH 2010 Talks*, SIGGRAPH '10, pages 26:1–26:1, New York, NY, USA, 2010.

[5] Mentor Graphics Corporation. *ModelSim Reference Manual*, 2010.

[6] A. Okazaki, S. Tsunekawa, T. Kondo, K. Mori, and E. Kawamoto. An automatic circuit diagram reader with loop-structure-based symbol recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 10(3):331–341.

[7] M. Tiwari, H. M. Wassel, B. Mazloom, S. Mysore, F. T. Chong, and T. Sherwood. Complete information flow tracking from the gates up. In *ASPLOS '09*, pages 109–120, 2009.

[8] M. Q. Wang Baldonado, A. Woodruff, and A. Kuchinsky. Guidelines for using multiple views in information visualization. AVI '00, pages 110–119, New York, NY, USA, 2000.

Memristors for Neural Branch Prediction: A Case Study in Strict Latency and Write Endurance Challenges

Hebatallah Saadeldeen, Diana Franklin, Timothy Sherwood, Frederic T. Chong
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, USA
Email: $\{heba, franklin, sherwood, chong\}$ @$cs.ucsb.edu$

Dmitri Strukov
Electrical and Computer Engineering
University of California, Santa Barbara
Santa Barbara, USA
Email: strukov@ece.ucsb.edu

***Abstract*—Memristors offer many potential advantages over more traditional memory technologies, including the potential for extreme densities, and fast read times. Current devices, however, are plagued by problems of yield, and durability. We present a limit study of an aggressive neural network application that has a high update rate and a strict latency requirement, analog neural branch predictor. Traditional analog neural network (ANN) implementations of branch predictors are not built with the idea that the underlying bits are likely to fail due to both manufacturing and wear-out issues. Without some careful precautions, a direct one-to-one replacement will result in poor behavior.**

We propose a hybrid system that uses SRAM front-end cache, and a distributed-sum scheme to overcome memristors' limitations. Our design can leverage devices with even modest durability (surviving only hours of continuous switching) to provide a system lasting 5 or more years of continuous operation. In addition, these schemes allow for a fault-tolerant design as well. We find that, while a neural predictor benefits from larger density, current technology parameters do not allow high dense, energy-efficient design. Thus, we discuss a range of plausible memristor characteristics that would; as the technology advances; make them practical for our application.

I. Introduction

As concerns mount about the end of DRAM scaling, researchers are examining alternative memory technologies and the new applications they enable. Resistive devices based on transition metal oxide (e.g. Memristors) could potentially replace typical memories like Flash, DRAM and perhaps SRAM. In fact, a 32Gb memristor crossbar array has been recently demonstrated [1].

Among the most important advantages of Memristors are high density, analog properties [2], CMOS-compatibility and possibility of monolithical 3D stacking [3]. These properties enable a wide range of potential applications ranging from memory to computations, from digital circuits to analog circuits, and including nonvolatile memory, signal processing circuits and artificial neuromorphic networks.

Specifically, Memristors and neural applications are always thought to be a perfect match. The strong affinity between Memristors and neural networks is generally based upon the assumption that memristors would be used to create an analog network of neurons. Unfortunately, this purely analog implementation might be impractical given the serious issues that currently remain with the technology, such as *low write endurance* (each cell can only be written 10^{12} times [4]) and *high percentage of defective devices* (≈10% defective devices [5]). We investigate this assumption given the recent milestones in the technology with the goal of guiding future technology advancements in order to be valuable for real applications.

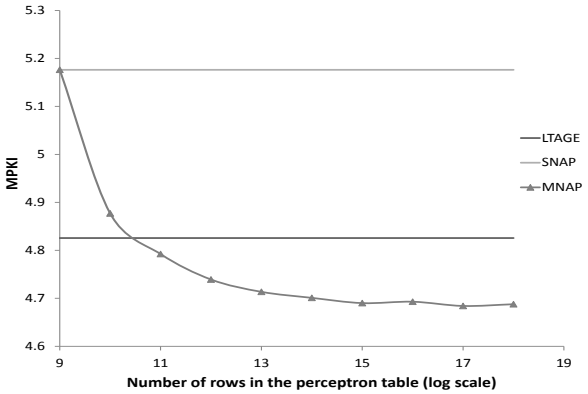


Fig. 1: Misprediction Per Kilo Instructions (MPKI) for SPEC CPU as the size of the perceptron table increase from 2^9 rows to 2^{18} rows. SNAP and LTAGE are the most up to date predictors. At 2^{13} (≈20X density improvement over SRAM), accuracy improves by 9% over SNAP, and 3% over LTAGE.

For our study, we choose an analog neural branch predictor that is derived from the perceptron branch predictor [6]. Neural predictors are one of the most accurate up to date predictors for SPEC traces. Further improving their accuracy would improve the overall performance of the applications by avoiding mispeculation, i.e., executing instructions on the wrong path. This application was chosen for the following reasons. First, it benefits from Memristors' advantages of *high density* to improve the accuracy of prediction as shown in figure 1. Second, it benefits from Memristors' ability to do *analog computations* efficiently which is the bottleneck operation for neural applications in general. Finally, it serves as a limit study for memristor technology due to its strict latency requirement (prediction within a single cycle) and frequent updates ($<$ microsec).

We propose general techniques to overcome write endurance limits, and provide fault tolerance that are necessary

for Memristors in order to be valuable for real applications without sacrificing accuracy. Our techniques are specifically valuable for latency strict applications. In order to evaluate the effectiveness of our schemes, we used both trace-driven simulations (infrastructure provided by CBP-2 [7]) and cycle-accurate simulations (PolyScalar). Trace-driven simulations are used to explore the design space for our techniques, e.g., determine the optimal table sizes, cache sizes, and number of instances. Using these parameters, cycle-accurate simulation is used to provide an overall performance evaluation. Our techniques are discussed in the next section.

II. Mitigating Challenges

We propose architecture solutions that address Memristors' challenges while taking advantage of their high density and ability to do analog computations.

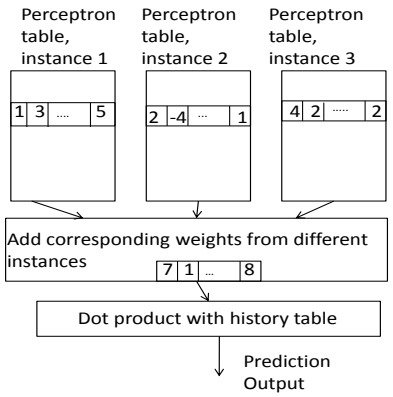


Fig. 2: Prediction Computation in case of a Cache miss

1) Hybrid Perceptron Table: An SRAM cache is used to keep the most recently *updated* table entries. This affects both reads and writes to the memristor perceptron table. On a read, the SRAM cache and memristors are read in parallel. In case of a cache hit, cache value is used o.w. Memristor value is used. On writes, if the entry is in the cache, it is written there. If not, the entry is first moved to the cache, and then written. This may evict a cache line, necessitating a write to the Memristors. We performed space design exploration to determine the optimal size of the cache which was found to be 12KB from 32KB hardware budget. Caching extends the lifetime of the predictor running SPEC CPU traces from 4 months to 24 months. Thus, caching is beneficial but is not enough to ensure a 5-year lifetime.

2) Distributed Sum: Many instances of Memristor are used to store a single weight value. When an update occurs for this value, only one instance is updated, increasing the lifetime of the value. These instances are organized such that the final weight value is the sum of values across all instances. Thus, assuming N writes to a weight, R instances and writes are evenly distributed among all instances, N/R writes is performed to each instance of the weight.

On memristor read, the instances values are added before they are used. In order to store a value, first an instance to modify is chosen. The old value is read; which is the sum of all instance values masking the entry of the instance to be modified. The delta of the old and the new value is calculated, and then written to the candidate instance which is chosen using global round- robin scheme (GRR). GRR uses only one counter for the entire table. Figure 2 shows how this algorithm is applied to perform prediction computations assuming three instances of the predictor table.

The main advantage of our scheme is that it exploits memristors' unique analog properties that make predictions within a 1ns feasible. We found that using 20 instances, in addition to caching, would extend the lifetime of the predictor to more than 5 years as well as make it fault-tolerant to 10% defective devices.

III. Discussion

In order for the considered application to benefit from the use of Memristors while overcoming its major limitations, it requires extreme density ($\approx$ 400X more dense than SRAM) while maintaining low access latency. As of today, memristor device properties are not good enough for the considered application. According to a recent study done by Xu et al. [8], 8MB Memristor crossbar memories would have 1.7ns read latency but only provides density advantage of $\approx$ 6x over SRAM memory, which falls several magnitude short of the number required to start getting performance benefits. However, these results are directly related to both the devices' assumptions as well as the peripheral logic. Thus, assuming more aggressive assumptions, which are nevertheless reasonable and based on the understanding of the physics behind memristor operation, could significantly improve memory performance.

For example, improving the density could be achieved through the following. First, the crossbar wire size and pitch can be scaled very aggressively down to 5 nm e.g. using nanoimprint technology [9]. Second, multiple layers can be stacked monolithically - effectively reducing even further the footprint of the devices. Finally, storing multiple bits per memristor cell [2]. Thus, assuming smaller crossbar pitch, a near term 100x density advantage of the crossbar Memristor memory can be achieved with either $F_{nano} \approx$ 13nm . Or alternatively, with F_{nano} = 30nm and 5 monolithically stacked crossbar layers. Over the long term, a 1000x density could be obtained with F_{nano} = 13nm and 10 crossbar layers [3].

References

[1] T.-Y. Liu, T. H. Yan *et al.*, "A 130.7 mm2 two-layer 32-gbit reram memory device in 24-nm technology," in *ISSCC*, 2013.

[2] B. H. F. Alibart, L. Gao and D. Strukov, "High-precision tuning of state for memristive devices by adaptable variation-tolerant algorithm," in *Nanotechnology*, 2012.

[3] D. Strukov *et al.*, "Four-dimensional address topology for circuits with stacked multilayer crossbar arrays," *Proc. of NAS.*, 2009.

[4] M.-J. Lee *et al.*, "A fast, high-endurance and scalable non-volatile memory device made from asymmetric Ta205-x/TaO2-x bilayer structures," *Nature Materials*, 2011.

[5] S. H. Jo, K.-H. Kim, and W. Lu, "High-density crossbar arrays based on a si memristive system," *Nano Lett.*, vol. 9, no. 2, pp. 870–874, 2009.

[6] D. A. Jimenez and C. Lin, "Dynamic branch prediction with perceptrons," *HPCA*, vol. 0, 2001.

[7] *JILP Special Issue: The Second Championship Branch Prediction Competition (CBP-2)*, 2007, http://cava.cs.utsa.edu/camino/cbp2.

[8] C. Xu, X. Dong, N. Jouppi, and Y. Xie, "Design implications of memristor-based rram cross-point structures," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, march 2011.

[9] W. Wu *et al.*, "Sub-10 nm nanoimprint lithography by wafer bowing," *Nano Letters*, 2008.

Fast Clustering Methods for Genetic Mapping

Veronika Strnadova, John Gilbert
University of California, Santa Barbara

Aydin Buluc, Leonid Oliker
Lawrence Berkeley National Laboratory

Joseph Gonzalez, Stefanie Jegelka
University of California, Berkeley

Jarrod Chapman, Daniel Rokhsar
Joint Genome Institute

***Abstract*—State-of-the-art genetic mapping tools, vital in the construction of complicated plant genomes, have not been able to keep up with the scale of genetic marker data currently available. Because these tools rely on a clustering algorithm quadratic in the number of markers in the linkage group-finding phase of genetic mapping, they cannot make use of the enormous datasets being generated by new sequencing technologies. While remaining competitive with these tools in terms of clustering quality, we propose new, more efficient clustering methods for genetic mapping.**

I. Introduction

Graph-analytic methods are increasingly finding their way into the domain of genomic science. Methods for finding a minimum spanning tree, motif finding, and clustering are helping geneticists effectively utilize the large scale and fine granularity of genomic sequence data. Next generation sequencing technologies have produced a flood of information which could prove highly beneficial to genetic map construction as well as *de novo* assembly, provided that there exist good graph clustering methods tailored to genomic data. In this work, we strive to evaluate and develop efficient and accurate graph clustering algorithms for genomics applications, which seek to identify linkage groups from a large dataset of genetic markers.

II. Background

The advent of low-cost, high throughput "next generation" sequencing technologies has brought about a host of new opportunities for genomics researchers. Millions of long *reads*, consisting of hundreds of base pairs, provide the potential for a fast reassembly of a DNA sample. Next generation sequencing allows geneticists to produce millions of genetic markers, the key ingredients to an accurate, high resolution genetic map. The map, in turn, provides a detailed blueprint of a true genomic sequence [3].

However, a computational challenge still exists in the automatic construction of genetic maps from genetic marker information. Current software tools, such as the popular MSTMap [4] and JoinMap [5], fail to scale up to even tens of thousands of markers, much less the millions of markers made available for analysis by next generation sequencing technologies. The bottleneck in these tools is the "grouping" stage of the genetic map construction pipeline, which finds linkage groups by computing pairwise marker similarity. Fig. 1 offers a simple diagram of this pipeline. In the first stage of genetic mapping, markers are clustered into linkage groups, based on a similarity metric which depends on the probability of recombination events between markers [3],[4]. Traditional mapping tools employ simple clustering algorithms, taking $O(m^2)$ time or more to cluster m markers. In our ongoing work, we aim to remove this handicap by introducing a fast clustering algorithm for genetic map construction. Our end-goals are threefold: 1) provide a power iteration-based clustering method tailored for linkage group finding, 2) devise a fast linkage group-finding algorithm based on assumptions and intutitions about the nature of genetic marker datasets, and 3) discover which of these approaches is best-suited for the genetic mapping problem.

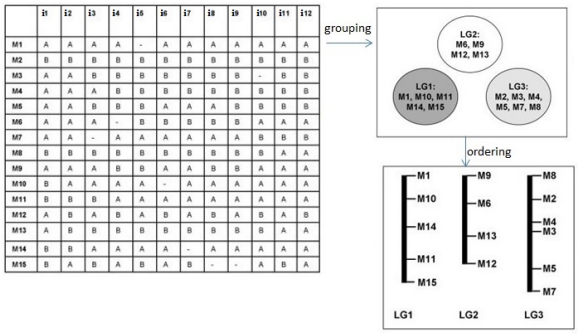


Fig. 1: A pipeline illustrating genetic map construction. Input is given in the form of a marker (M)-individual (i) matrix, indicating the genotype of each individual in a selected population at each marker site. In the first "grouping" stage, markers are clustered into linkage groups (LG), which often correspond to physical chromosomes. In the next stage, markers within each linkage group are ordered, giving a blueprint for the sequence of the entire genome.

III. Algorithmic Overview

We now give an overview of the algorithms that we are developing and testing for use in the grouping stage of genetic mapping construction tools.

A. Power Iteration Clustering

Recently, a new clustering algorithm based on power iteration (PI) was introduced to the machine learning community [1],[2]. Lin and Cohen's "Power Iteration Clustering (PIC)"[1] deviates from the traditional power iteration algorithm [1] by introducing a new convergance criterion: the PIC algorithm stops iterating when the *acceleration*, i.e. the rate of change

[1]Recall that power iteration simply multiplies a matrix A by a vector v, then normalizes this product, until the difference in two iterations of this process is very small. In other words, we have at each step $v^{t+1} = \frac{Av^t}{\|Av^t\|}$ until $v^{t+1} - v^t$ is close to 0. v^t converges to the principal eigenvector of A

of the rate of change, of the iteration vector is close to 0. Each element v_i^{final} of the converged iteration vector v^{final}, what is known as a "pseudo-eigenvector", will lie close to one of k distinct values, assuming the data is separable into k categories. This reflects the assumption that if k eigenvectors are sufficient to cluster a dataset with spectral clustering, then the first k eigenvalues should be well-separated from the k+1st to nth eigenvalues. Following this logic, the final stage of PIC clusters the elements of v^{final} using k-means in 1 dimension.

Following Lin and Cohen's method for fast clustering of text datasets [2], we attempt to avoid constructing the similarity matrix S that is used in each iteration of PI. We replace the product Sv with two sparse matrix-vector products $M(M^Tv)$. We note that this decomposition is not trivial, because the similarity between two markers is based on the nonlinear "LOD score", a logarithm of odds that two markers are genetically linked. Because the LOD score cannot be expressed as a simple inner product, we make use of a maximum likelihood estimation of the LOD score given by Wu et. al [4] to express the similarity matrix S as the sparse matrix-sparse matrix product $M(M^T)$.

Given that many optimization techniques exist for sparse linear algebraic operations, rewriting Sv as $M(M^Tv)$ provides significant speedup. In addition, we attempt to improve clustering quality by substituting the k-means algorithm used in the final stage of PIC with an improved version of k-means, called k-means++. We have also experimented with k-means++ in 2 dimensions, hoping that the extra computation required to find a second pseudo-eigenvector will be outweighed by gains in clustering quality.

B. Randomized Core Set Clustering

Although we believe in the power and efficiency of power iteration clustering for general clustering problems, we attempt to outperform this approach with a simpler algorithm by exploiting certain biological properties of our data. Our *randomized core set* (RCS) algorithm relies on the assumption that the structure of each cluster emulates that of a chromosome – i.e., markers are ordered, exhibiting a linear cluster structure. RCS processes markers sequentially and randomly, dynamically updating linkage groups as well as a "core set" of representative markers (*RM*'s) per linkage group. RM's are markers chosen such that each marker in a linkage group is within a threshold LOD of at least one RM.

At each iteration of RCS, a LOD score is computed for one marker and each of the RM's for each existing cluster. Then, one of three things happens: 1) the marker sprouts a new linkage group, becoming the sole RM for that linkage group, 2) the marker is assigned to an existing linkage group 3) the marker is assigned to an existing linkage group, and becomes a new RM for that group. Determining whether the marker falls into category (1), (2), or (3) above is a constant-time operation. If the cluster structure is indeed linear, then we expect the number of representative points r to be small, giving a runtime which is *close to linear* in the number of markers. This is a significant improvement over the quadratic algorithms currently in use in popular mapping tools.

TABLE I: F-measures Representing Clustering Qualtiy

Algorithm	overall F-measure
PIC with k-means	0.527957
PIC with k-means++	0.648037
PIC with 2D k-means++	0.724582
Randomized Core Set	0.989806

IV. Initial results & Conclusion

Researchers from the Joint Genome Institute (JGI) have generously provided us with a real dataset of 113,000 genetic markers, too large for any current mapping tool to handle in a reasonable amount of time. We have implemented the PIC algorithm with k-means, k-means++, and 2-dimensional k-means++ in the final clustering stage, as well as the randomized core set algorithm discussed in the previous section.

We use the F-measure, ranging from 0 to 1, to compare the clustering results of each of these algorithms to the set of clusters found by JGI researchers. The F-measure is a weighted harmonic mean of precision and recall [6], comparing all clusters in a clustered set to the "golden standard" clusters (in our case, the JGI set). An F-score of 1 for a particular clustering would indicate perfect precision and perfect recall for each of the k "golden standard" clusters. Table I shows the best F-score obtained thus far, for each of the clustering algorithms discussed above.

We are also testing our algorithms on simulated data ranging from 5,000 to 800,000 markers. These datasets were created using a simulation program which accounts for "real-life" problems in genetic mapping, allowing us to specify, among other possibilities, the amount of missing data and the error rate in sequencing the markers. Thus far, our randomized core set algorithm is outperforming both JoinMap and MSTMap in terms of efficiency, and is highly competitive in terms of F-scores, scoring higher than 0.95 in all cases tested to date.

Clearly, the addition of k-means++ as well as 2-dimensional final-stage clustering improve the quality of clusters obtained by the PIC-based methods. However, our randomized core set algorithm outperforms these sophisticated methods, indicating that the structure of each cluster is indeed nearly linear and can be efficiently exploited. We are hopeful that our efforts will lead to significant speedups in the genetic map construction process in the bright near future.

References

[1] F. Lin, W. Cohen. *Power Iteration Clustering.* In Proceedings of the 27th international conference on machine learning, pp.655-662. 2010

[2] F. Lin and W. Cohen. *A Very Fast Method for Clustering Big Text Datasets.* In Proceedings of the 19th European conference on artificial intelligence, pp.303-308. 2012

[3] J. Cheema, J. Dicks. *Computational approaches and software tools for genetic map estimation in plants.* In Briefings in Bioinformatics. Vol.10(6) pp.595-608. 2009

[4] Y. Wu, P. Bhat, T. Close, S. Lonardi. *Efficient and Accurate Construction of Genetic Linkage Maps from the Minimum Spanning Tree of a Graph*. PLoS Genetics Vol 4(10). 2008

[5] P. Stam. *Construction of integrated genetic linkage maps by means of a new computer package: Join Map*. The Plant Journal, Vol.3(5). 1993

[6] S. Wagner, D. Wagner. *Comparing Clusterings - An Overview*. 2007

A Simple, Efficient Preconditioner for Graph Laplacians

Erik G. Boman
Scalable Algorithms Dept.
Sandia National Laboratories, NM*

Kevin Deweese†
Dept. of Computer Science
UC Santa Barbara

***Abstract*—We consider the solution of linear systems corresponding to the combinatorial graph Laplacian of large, unstructured networks. Although several recent theoretical algorithms can solve such problems in near-linear time, there is a lack of practical algorithms and software. We discuss an implementation in Trilinos [1] of some simple graph preconditioners, tools for speeding up linear solvers. The preconditioners proposed belong to the class of support tree preconditioners. We show that a subgraph formed by multiple maximum spanning forests can be an effective preconditioner in practice.**

I. Introduction

Networks play an important role in many application areas, for example, engineering, social sciences and biology. We focus on networks that are large and unstructured. Several analysis techniques rely on the graph Laplacian. Solving linear systems or eigensystems for graph Laplacians can be a very compute intensive task. Preconditioners can dramatically reduce the solution time. Although there are many good preconditioners for PDEs, they are generally not suited for graph Laplacians from highly irregular graphs or scale-free networks. Over the last decade, several graph-based preconditioners with strong theoretical properties have been developed [2], [3]. However, these are typically too complicated to implement, so there is a lack of practical implementations and software. We seek to bridge this gap by studying the performance of some generalizations of Vaidya's spanning tree preconditioner [4], that are simple to implement.

A. Background

The combinatorial graph Laplacian of a graph G is given by $L_G = D - A$, where A is the adjacency matrix of G and D is a diagonal matrix containing the vertex degrees. L_G is positive semidefinite and diagonally dominant.

A good preconditioner M of L_G should reduce the number of iterations to solve the preconditioned system $M^{-1}L_G$ with an iterative solver. In addition solving the system $Mw = y$ should be much easier to solve than $L_G x = b$ as it will be solved at every iteration. The first constraint prompts us to bound the condition number of the matrix while the second constraint requires us to bound the fill in the triangular factors of the preconditioner. Assuming a complete Cholesky factorization is used these factors will be of the form $M = LL^T$ and will be used to quickly solve $LL^T w = y$.

The maximum spanning tree preconditioner proposed by Vaidya has condition number $O(nm)$ which bounds the number of iterations of the preconditioned system to $O(\sqrt{nm})$, where n is the dimension of the matrix (number of vertices) and m is the number of non-zero entries in the matrix (number of edges). Typically the number of iterations required is much less than these worst case bounds. Since the preconditioner corresponds to a tree, it can be factored with no fill. This is a simple preconditioner that is fast to compute, but convergence can be quite slow. Much work has focused on *graph sparsification* to find better subgraphs for preconditioning.

II. The New Preconditioner

A. Maximum Spanning Forest

There has been much work on how to modify the spanning tree to yield a better approximation of the original graph. One technique is to carefully select edges to add that both improve the quality and only slightly increase the cost of factoring the preconditioner. Another approach is to solve for the preconditioner recursively, which eliminates the need to factor it with little fill. However, this makes a solver implementation quite complex and prevents the use of standard scientific software libraries.

We propose to approximate a graph with k spanning forests, where $k \geq 1$ is a parameter. MSF(1) is simply the maximum-weight spanning tree. For $k > 1$, we remove the tree edges from the original graph and find a spanning forest of the remaining graph. We repeat this process k times, and the union of all the spanning forests is our MSF(k) preconditioner. The term forest is used since removing a tree could leave the graph disconnected. One drawback of the method is that we do not limit the fill in the Cholesky factors, so memory usage may be large. However, we observe that in practice it is quite reasonable. A promising modification is to perform an incomplete Cholesky (IC) factorization of the preconditioner which will drop some entries during factorization yielding a sparser approximate factorization $M \simeq LL^T$.

B. Initial Results

We have implemented the MSF(k) method in Trilinos, and it is publicly available. We illustrate the effectiveness of our method by an example. We solve for the graph Laplacian of

*Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energys National Nuclear Security Administration under contract DE-AC04-94AL85000.

†Work supported by grants from Intel, Lawrence Berkeley National Lab, and Microsoft.

Preconditioner	Iterations	Solve Time (s)
None	528	1.08
Jacobi	239	0.917
MSF(1)	203	0.820
MSF(2)	132	0.646
MSF(2) + IC	186	0.654

TABLE I: hvdc1: Time and iterations to solve.

Preconditioner	Iterations	Solve Time (s)
None	791	26.05
Jacobi	322	21.31
MSF(1)	251	16.84
MSF(2)	156	17.73
MSF(2) + IC	226	12.60

TABLE II: pa2010: Time and iterations to solve.

the hvdc1 graph/matrix from the UF collection [5], which has 24,842 vertices and 79,213 edges. We use the preconditioned conjugate gradient solver from the Belos package in Trilinos and solve for a residual tolerance of 10^{-6}. We see from Table I that the number of iterations and run times are significantly lower than for non-preconditioned or Jacobi. As expected, the number of iterations decreases after adding an additional forest. The denser factors of MSF(2) mean that the preconditioner is more expensive to use at every iteration but in this example the trade-off is worth it as the decrease in the number of iterations leads to MSF(2) having the best solve time*. To deal with the problem of possible fill in the factor the MSF(2) preconditioner was also factored incompletely with no fill. We can see that the iteration count increases as this is a worse approximation. In this example it is not worth factoring incompletely. There are matrices where it was found to be useful. For example the results of the pa2010 graph/matrix of the UF collection, which has 421,545 vertices and 1,029,231 edges, is shown in Table II. When MSF(2) is factored incompletely applying the sparser preconditioner causes that the solve time to improve, even with more iterations. MSF(2) by itself is not an improvement over MSF(1) in this case due to the extra fill in the preconditioner. This second example seems to be more typical of the graphs/matrices we have tested so far. The solve times of each preconditioner on 13 unstructured graphs/matrices from the UF collection are shown in Figure 1. Each vertical set of 3 data points are experiments run on one of the graphs. MSF(2) factored completely seems to be too expensive but MSF(2) factored incompletely appears to be a promising competitor to MSF(1).

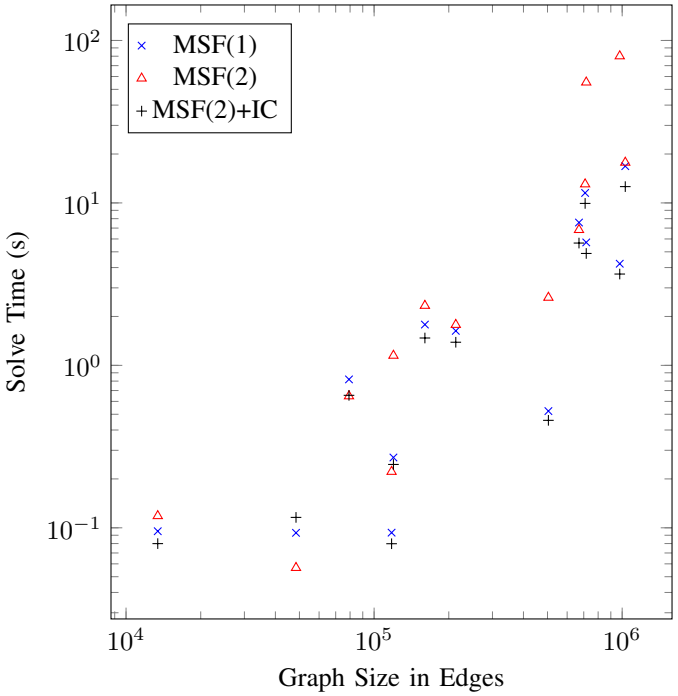


Fig. 1: Comparison of solve times

III. Discussion and Conclusion

Adding an additional spanning forest can improve performance but it seems that an incomplete factorization will be needed to reduce fill in the preconditioner factors. There are numerous areas of future experimentation with the current Trilinos implementation. MSF(2)† could be factored with a variety of flavors and parameters of IC to try and find a better trade-off between preconditioner quality and cost of application, hopefully yielding a more impressive performance improvement. In addition further experimentation should be done to try and understand which classes of graphs benefit from these types of preconditioners.

*Note that we did not include the preconditioner setup time, the time to find the spanning forest, or the time to factor the preconditioner since these costs can often be amortized over many solves.

†In the graphs we experimented with it was never worth adding yet another spanning forest, especially since by that point most of the original edges were already added back to the preconditioner

Acknowledgment

The authors would like to thank John Gilbert.

References

[1] M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, A. G. Salinger, H. K. Thornquist, R. S. Tuminaro, J. M. Willenbring, A. Williams, and K. S. Stanley, "An overview of the trilinos project," *ACM Trans. Math. Softw.*, vol. 31, no. 3, pp. 397–423, Sep. 2005. [Online]. Available: http://doi.acm.org/10.1145/1089014.1089021

[2] D. A. Spielman and S.-H. Teng, "Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems," in *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, ser. STOC '04. New York, NY, USA: ACM, 2004, pp. 81–90. [Online]. Available: http://doi.acm.org/10.1145/1007352.1007372

[3] I. Koutis, G. Miller, and R. Peng, "Approaching optimality for solving SDD linear systems," in *Proc. of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, 2010, pp. 235–244.

[4] M. Bern, J. R. Gilbert, B. Hendrickson, N. Nguyen, and S. Toledo, "Support-graph preconditioners," *SIAM J. on Matrix Anal. and Appl.*, vol. 27, no. 4, pp. 930–951, 2006.

[5] T. A. Davis and Y. Hu, "The University of Florida Sparse Matrix Collection," *ACM Transactions on Mathematical Software*, vol. 38, no. 1, pp. 1:1–1:25, Nov. 2011. [Online]. Available: http://doi.acm.org/10.1145/2049662.2049663

On the Most Likely Convex Hull of Uncertain Points in the Plane

Hakan Yıldız

Department of Computer Science
University of California, Santa Barbara
Santa Barbara, California 93106
Email: hakan@cs.ucsb.edu

***Abstract*—Consider a set of points in the plane, where the existence each point is uncertain and only known probabilistically. We study the problem of finding the *most likely convex hull*, i.e., the polygon with the highest probability of being the convex hull of the points. We present an $O(n^3)$ dynamic programming algorithm to solve the problem.**

I. Introduction

Given a set of points $\mathcal{P}$ in the plane, the *convex hull* of $\mathcal{P}$ is a minimal convex polygon that contains all points in $\mathcal{P}$. Computing convex hulls is a well-studied problem and worst-case optimal algorithms exist for both its planar and high-dimensional forms [3].

In this work, we revisit the convex hull problem in an uncertainty setting where the existence of the input points is uncertain and only known probabilistically. In particular, we are given a point set $\mathcal{P}$ in the plane such that each point of $\mathcal{P}$ is known to exist with an independent probability. Under these conditions, we want to compute a convex polygon C such that the probability that C is the convex hull of the outcome of a probabilistic experiment is maximum. For simplicity, we refer to this polygon as the *most likely convex hull* of $\mathcal{P}$.

In addition to the theoretical motivation, the most likely convex hull problem is also motivated by applications that deal with uncertain data. For instance, in *movement ecology* [5], [6], one method frequently used to visualize natural habitats for animals is to compute the convex hull of the locations where the animal is observed. However, it has been long known by ecologists that this method often produces a gross over-estimation of the actual habitat, due to outlier locations. One way to overcome this issue is to use a discrete landmarks, each of which is assigned a probability (based on the frequency of visits of the animal to its vicinity), and then use the most likely hull of the landmarks as the most probable natural habitat.

Our contribution is a dynamic programming algorithm to compute the most likely convex hull in $O(n^3)$ time and $O(n^2)$ space. We note that our method resembles some similar approaches previously proposed for various convex subset problems [1], [2].

II. Definitions

Let $\mathcal{P}$ be an uncertain set of n points. We represent $\mathcal{P}$ as a set of pairs $\{(s_1, \pi_1), \ldots, (s_n, \pi_n)\}$ where each s_i is a point in the plane and each π_i is a real number in the range $(0, 1]$. The point s_i is the *site* where the ith point of $\mathcal{P}$ may exist, and the number π_i is the associated probability of existence. In a probabilistic experiment, the ith point exists at s_i with probability π_i and is absent otherwise.

Let S be the set of all sites for the points in $\mathcal{P}$, i.e., $S = \{s_1, \ldots, s_n\}$. A subset $A \subseteq S$ is the outcome of a probabilistic experiment on $\mathcal{P}$ with probability:

$$\prod_{i \mid s_i \in A} \pi_i \times \prod_{i \mid s_i \notin A} \overline{\pi_i}$$

where $\overline{\pi_i}$ is the complementary probability $(1 - \pi_i)$. For ease of reference, we denote this probability by $\pi(A)$. Consider a convex polygon C. We define the *likeliness* of C, denoted $\mathcal{L}(C)$, to be the probability that C is the convex hull of the outcome of a probabilistic experiment on $\mathcal{P}$. We can write $\mathcal{L}(C)$ as

$$\mathcal{L}(C) = \Pr\big[\, \mathcal{CH}(A) = C \,\big] = \sum_{\substack{A \subseteq S \\ \mathcal{CH}(A) = C}} \pi(A)$$

where $\mathcal{CH}(A)$ is the convex hull of A. The *most likely convex hull* of $\mathcal{P}$ is the polygon C that maximizes $\mathcal{L}(C)$.

III. Algorithm

We now explain our algorithm to find the most likely convex hull. We begin with an observation. Let C be a convex polygon and V be the set of its vertices such that $V \subseteq S$. Let $S_{out} \subseteq S$ be the set of sites outside C. Observe that C is the convex hull of an experiment outcome $A \subseteq S$ if and only if A contains all points in V and no points in S_{out}. This implies the following relation, whose formal proof we omit from this abstract:

Lemma 1.

$$\begin{aligned} \mathcal{L}(C) &= \Pr\big[\, V \subseteq A \ \wedge \ S_{out} \cap A = \emptyset \,\big] \\ &= \prod_{i \mid s_i \in V} \pi_i \times \prod_{i \mid s_i \in S_{out}} \overline{\pi_i} \end{aligned}$$

For each site $s_i \subseteq S$, we compute the most likely convex hull C such that s_i is the lowest vertex (along y-axis) of C. The hull with the maximum likeliness among these is the most likely convex hull of $\mathcal{P}$. We compute the most likely hull for a particular lowest vertex s_i as follows. Let s_j and s_k be two sites in S above s_i such that s_k is radially to the left of s_j

with respect to s_i. Let G_j^k denote open region bounded by the segment $s_j s_k$ and the rays $\overrightarrow{s_i s_j}$ and $\overrightarrow{s_i s_k}$. (See Figure 1a.) We define the *contribution* of the directed edge $s_j\vec{s}_k$, denoted $\mathcal{C}(s_j\vec{s}_k)$, as π_j times the product of the complementary probabilities of all sites in G_j^k. That is,

$$\mathcal{C}(s_j\vec{s}_k) = \pi_j \times \prod_{u \mid s_u \in G_j^k} \overline{\pi_u}$$

For a site s_j above s_i, let G_i^j to be the open region bounded by the downward ray extending from s_i and the ray $\overrightarrow{s_i s_j}$. We define G_j^i be the complementary region of G_i^j. (See Figure 1b.) We define the contribution of the directed edge $s_i\vec{s}_j$, denoted $\mathcal{C}(s_i\vec{s}_j)$, as π_i times the product of the complementary probabilities of all sites in G_i^j. Similarly, we define the contribution of the reverse edge $s_j\vec{s}_i$, denoted $\mathcal{C}(s_j\vec{s}_i)$, to be π_j times the complementary probabilities of all sites in G_j^i. The following lemma shows how edge contributions relate to the likeliness of a convex polygon.

Lemma 2. *Let C be a convex polygon with vertices $s_i, s_{\alpha(1)}, \ldots, s_{\alpha(m)}$ in counter-clockwise order such that s_i is the lowest vertex of C. Then,*

$$\mathcal{L}(C) = \mathcal{C}(s_i\vec{s}_{\alpha(1)}) \times \mathcal{C}(s_{\alpha(1)}\vec{s}_{\alpha(2)}) \times \cdots \times \mathcal{C}(s_{\alpha(m-1)}\vec{s}_{\alpha(m)}) \times \mathcal{C}(s_{\alpha(m)}\vec{s}_i)$$

Proof: One can partition the space outside C into the regions $G_i^{\alpha(1)}, G_{\alpha(1)}^{\alpha(2)}, \ldots, G_{\alpha(m-1)}^{\alpha(m)}, G_{\alpha(m)}^i$ by drawing a downward ray from s_i and drawing rays $\overrightarrow{s_i s_{\alpha(j)}}$ for each $1 \le j \le m$. (See Figure 1c for an example.) Then, by Lemma 1, it is easy to see that the $\mathcal{L}(C)$ is the product of the contributions of the edges of C. ∎

We note that the contribution of each edge can be computed in constant time after an $O(n^2)$-time preprocessing. The idea in brief is to utilize a modified version of a triangle query structure by Eppstein et al. [4]. We omit the details of this structure from this manuscript.

Our dynamic programming strategy is as follows. For each directed edge $s_j\vec{s}_k$, we compute the convex chain starting at s_i and ending with edge $s_j\vec{s}_k$, such that the product of contributions of the edges participating in the chain is maximum. Let $\mathcal{T}(s_j\vec{s}_k)$ denote the product of contributions for this chain (and also the chain itself as an abuse of notation). By Lemma 2, the chain $\mathcal{T}(s_j\vec{s}_i)$ with the maximum value among all sites s_j is the most likely convex hull whose lowest vertex is s_i. The following recurrence is easy to observe and is the core of our dynamic programming algorithm:

$$\mathcal{T}(s_j\vec{s}_k) = \mathcal{C}(s_j\vec{s}_k) \times \max_{\substack{s \in S \\ s \text{ is radially right to } s_j \text{ w.r.t. } s_i \\ s \to s_j \to s_k \text{ is a left turn}}} \mathcal{T}(s\vec{s}_j)$$

We utilize the above recurrence to compute all $\mathcal{T}(\cdot)$ values as follows. We begin by setting $\mathcal{T}(s_i\vec{s}_j)$ to $\mathcal{C}(s_i\vec{s}_j)$ for all sites s_j. Then, we process all sites above s_i in right-to-left radial order (with respect to s_i). When we process a site s_j, we compute $\mathcal{T}(s_j\vec{s}_k)$ values for all sites s_k radially to the left of s_j. This can be done in $O(n)$ time as follows. Let L_j be the set of sites in S that are radially left to s_j plus s_i. Similarly, let R_j be the set of sites radially right to s_j plus s_i. Let $s_{\beta(1)}, \ldots, s_{\beta(m)}$ be the sites in R_j in clockwise order around s_j. For each site $s_{\beta(u)}$ in R_j, we define s_u^* be the site s among the sequence $s_{\beta(u)}, s_{\beta(u+1)}, \ldots, s_{\beta(m)}$ that maximizes $\mathcal{T}(s\vec{s}_j)$. The site s_u^* can be computed for all sites $s_{\beta(u)}$ in linear time by sweeping the sites in R_j in counter-clockwise order. (We compute this counter-clockwise order in a preprocessing step.)

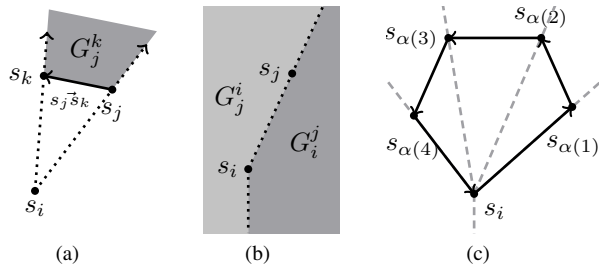


Fig. 1

For each site $s_{\beta(u)}$ in R_j, we set the value $\mathcal{T}(s_j\vec{s}_k)$ to $\mathcal{C}(s_j\vec{s}_k) \times \mathcal{T}(s_u^*\vec{s}_j)$ for all sites s_k in L_j inside the wedge bounded by the lines $\overleftrightarrow{s_{\beta(u-1)}s_j}$ and $\overleftrightarrow{s_{\beta(u)}s_j}$. Notice that the sites in this wedge are the sites that form a left turn when connected to $s_{\beta(u)}, s_{\beta(u+1)}, \ldots, s_{\beta(m)}$ through s_j (which is the condition in the recurrence relation). Finally, we note that by considering the sites $s_{\beta(u)}$ in radial order around s_j, we can locate each site in the wedge of interest in constant time.

The processing of a single point s_j takes $O(n)$ time, and thus we can find the most likely hull where s_i is the lowest vertex in $O(n^2)$ time. It follows that the most likely hull of $\mathcal{P}$ can be found in $O(n^3)$ time. The space cost is $O(n^2)$, dominated by the storage of the $\mathcal{T}(\cdot)$ values.

Theorem 1. *The most likely convex hull of n uncertain points in the plane can be computed in $O(n^3)$ time and $O(n^2)$ space.*

REFERENCES

[1] C. Bautista-Santiago, J. M. Díaz-Báñez, D. Lara, P. Pérez-Lantero, J. Urrutia, and I. Ventura. Computing optimal islands. *Operations Research Letters*, 39(4):246–251, 2011.

[2] V. Chvátal and G. Klincsek. Finding largest convex subsets. *Congresus Numeratium*, 29:453–460, 1980.

[3] M. De Berg, O. Cheong, M. Van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 2008.

[4] D. Eppstein, M. Overmars, G. Rote, and G. Woeginger. Finding minimum area k-gons. *Discrete & Computational Geometry*, 7(1):45–58, 1992.

[5] W. M. Getz, S. Fortmann-Roe, P. C. Cross, A. J. Lyons, S. J. Ryan, and C. C. Wilmers. Locoh: Nonparameteric kernel methods for constructing home ranges and utilization distributions. *PLoS ONE*, 2(2), 02 2007.

[6] W. M. Getz and C. C. Wilmers. A local nearest-neighbor convex-hull construction of home ranges and utilization distributions. *Ecography*, 27(4):489–505, 2004.

Android at Bandon Bay: Low-Cost Environmental Monitoring and Event Detection Using Smartphones

Michael Nekrasov
Moment Lab
UC Santa Barbara
Santa Barbara, USA
mnekrasov@cs.ucsb.edu

Sirilak Chumkiew
COE for Eco-informatics
Walailak University
Nakhon Si Thammarat, Thailand
sirilak.chumkiew@gmail.com

Peter Shin
CLEOS Lab
UC San Diego
La Jolla, USA
pshinn@ucsd.edu

***Abstract*— Our work builds on open source technologies and standards to provide a system for real-time event detection in Bandon Bay, Thailand. Our system leverages the availability and versatility of mobile devices for effective low-cost monitoring in a region that is devoid of power and is prone to frequent cellular disruptions.**

Keywords*— *Android; Real-time Analysis; Environmental; Observing System; Water Quality; Event Detection; Mobile Computing;

I. MOTIVATION

Bandon Bay, Surathani province, is home to mussel, cockle, oyster, and shrimp farmers. In March 2011, severe rainfall in the southern region of Thailand caused an influx of freshwater and sediment into the bay. Cockle and oysters were suffocated by a thick layer of sediment. The surge of water forced shrimp out into the open ocean. [1] The impact on the region was massive, with over half a million people affected [2]. In particular, the aquaculture industry suffered immensely. The aim of this project is to provide a valuable service to the region by giving farmers and locals a resource for assessing the water quality in Bandon Bay, as well as providing a warning system against possibly treacherous environmental patterns.

This work is part of a larger collaborative project, which enables the use of mobile devices for communication and computation for environmental sensor networks in regions without significant infrastructure. The work is a partnership between the Center of Excellence for Eco-informatics at Walailak University, the University of California Santa Barbara, and the University of California San Diego. It brings computer scientists and biologists together for the development of technology aimed at studying Thailand's costal ecosystems.

II. GOALS

1. To provide a valuable service to the region by giving farmers and locals a resource for assessing the water quality in the bay.
2. To provide the tools to develop an early warning system to farmers and locals.
3. To use open source software and cost effective hardware to create a sustainable system that is affordable for use in developing regions.

III. SYSTEM OVERVIEW

A. *Real-Time Data Collection*

Before detection can take place, the site first requires a system for real-time data acquisition. This is not a trivial task because the environmental conditions in the bay require a low power device that is capable of robust wireless communication. While there are industry devices capable of this, such as the Campbell Scientific sensor suites, they are prohibitively expensive for developing nations and are heavily proprietary, impeding expansion and development. After testing several systems, including a low-cost netbook and a Raspberry Pi, we determined that a mobile device running open source software is a perfect candidate. An Android smartphone is inherently capable of cellular communication, has a backup battery for intermittent power loss, and has the computing power necessary for data ingestion and onboard processing from a multitude of meteorological and aquatic sensors.

Mobile devices lack a native interface for accessing ecological sensors, which typically communicate via RS-232 or SDI-12 using both digital and analog protocols for communication. As we did not want to implement a proprietary system, we decided to build upon the Open Source SensorPod software stack developed at UCSD [3] for interfacing with environmental sensors.

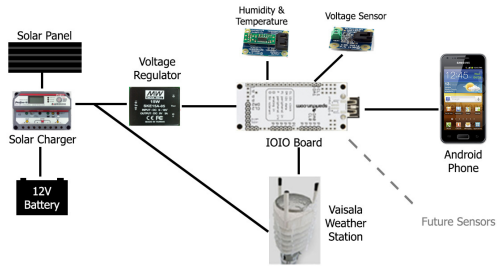


Fig. 1. Data Collection Platform

We recently deployed the system in the Gulf of Thailand, four kilometers off the coast. The system utilizes a Galaxy Nexus Android phone for real-time data collection and processing. For power, the system uses a 40W solar panel. The

data is transmitted using the phone's built in cellular modem. The system utilizes a SparkFun Electronics IOIO for interfacing to external sensors and power. The phone currently interfaces with a Vaisala WXT 520 meteorological station, with plans to extend the sensor suite to include water sensors for conductivity, dissolved oxygen, and pH.

B. *Real-Time Event-Detection*

The phone streams the data to an Amazon EC2 server utilizing the Open Source Data Turbine streaming middleware [4] for real-time buffered data streaming and visualization. This middleware is used internationally by many communities of ecologists as well as other disciplines requiring real-time data streaming. By building upon an existing system for reliable real-time data delivery, we wanted to allow interoperability with other tools and technologies developed by the ecological community. Through this approach our system can be used as infrastructure for future applications.

For event detection, the system integrates with Esper [5], a free and publically available complex event processing engine. Sensor streams captured via Data Turbine are sent through "SQL like" queries running on ESPER. When an event is detected, it activates a trigger, which marks this as an event of interest. In a later iteration of the system, an email, text message, or any other form of notification will be sent when a hazardous event is detected. By once again utilizing a freely available off the shelf system, Thai researchers can utilize the full power and feature set of a developed product, contributing to the sustainability and flexibility of the system.

While this iteration of the system runs at the server, ideally this detection code would run on the mobile device. Unfortunately, despite being written in Java, ESPER is currently only partially supported for Android and still remains unsuitable for this application. We hope that there will be an official Android version of ESPER or a comparable event processing engine, as there is already considerable interest.

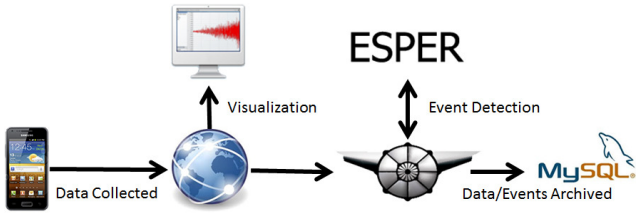


Fig. 2. Data Flow Schematic

IV. Results

We built on existing technologies to develop a novel system for real-time environmental monitoring of Bandon Bay. A mobile device collects and streams the data to a cloud server. The data is then run through a real-time event detection engine. Both the original data and the derived analysis are then made publically accessible and mirrored to universities in Thailand in real-time, using the DataTurbine middleware. Interested parties can visualize both the data and the derived analysis in real-time. The system utilizes solar power, and is tolerant of network disruption.

This system leverages the availability and versatility of mobile devices for low-cost effective monitoring. Thailand is a developing nation and cost plays a major factor in the feasibility of a system. Our monitoring solution utilizes freely accessible software (open source whenever possible) and off the shelf hardware requiring only minor customization to create a system that is not only powerful but affordable to deploy and maintain. As our system continues to grow and develop, we hope that it enables sustained automated monitoring and a platform for disaster detection in a critical region of a developing nation.

V. Future Work

Our work has sparked interest from other groups who are using common technologies for data collection. At a workshop at the start of July 2013, we met with groups from Taiwan (Forestry, Agriculture, Endemic Study), as well as engineers from NECTEC Thailand, and the University of Wisconsin Madison. We are exploring possibilities of adopting our work to other sites for using mobile devices for the automated detection of environmental factors.

Acknowledgment

The work in Thailand was funded by a Fulbright Scholarship spanning January to July 2013 by the United States Department of State. The work utilizes technologies from multiple projects, and we are especially grateful for the software and hardware developed by the CLEOS lab at UC San Diego. The collaboration between groups from different fields and countries builds on previous partnerships funded by grants from both NSF and the Gordon and Betty Moore Foundation.

References

[1] "Thailand flood death toll rises as rains ease." *BBC* April 05 2011. Web. 12 Feb. 2013.

[2] "Flood Summary in Southern Thailand 2011" *Asia Disaster Reduction Center (ASRC)* April 11 2011. Web. 12 Feb. 2013.

[3] Fountain, Tony, et al. "The open source Dataturbine initiative: empowering the scientific community with streaming data middleware." Bulletin of the Ecological Society of America 93.3 (2012): 242-252.

[4] Fountain, Tony, et al. "Digital Moorea cyberinfrastructure for coral reef monitoring." Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2009 5th International Conference on. IEEE, 2009.

[5] EsperTech Event Stream Intelligence. 12 Feb, 2013, http://www.espertech.com.

You are How You Click: Sybil Detection Using Clickstream Models

Gang Wang, Tristan Konolige, Christo Wilson†, Xiao Wang‡,
Haitao Zheng and Ben Y. Zhao
UC Santa Barbara †Northeastern University ‡Renren Inc.
{gangw, tkonolige, htzheng, ravenben}@cs.ucsb.edu, cbw@ccs.neu.edu, xiao.wang@renren-inc.com

***Abstract*—Fake identities (or *Sybils*) are pervasive in today's online communities who are responsible for a growing number of threats, including fake product reviews, malware and spam on social networks. In this paper, we propose a novel Sybil detection system using server-side clickstream models. By grouping "similar" user clickstreams into behavior clusters, we can capture Sybils whose behavior patterns deviate from normal users. We evaluate our system with a large ground-truth dataset and demonstrate its high accuracy. Currently, we work with our collaborators at Renren and LinkedIn to test our prototype.**

I. Introduction

It is easier than ever to create fake identities (Sybil accounts) in today's online social networks. Despite the increasing efforts from service providers, existing services cannot prevent attackers from creating large numbers of Sybil accounts. For example, Turing test based defenses such as CAPTCHAs are routinely solved by dedicated workers for pennies per request via crowdsourcing. This trend leads to a dramatic rise in forged and malicious online content such as fake reviews on Yelp, malware and spam on social networks [1], and large, Sybil-based political lobbying efforts.

Recent work has explored a number of potential solutions. Most focus on detecting Sybils in social networks by leveraging the assumption that Sybils will find it difficult to befriend real users. This forces Sybils to connect to each other and form strongly connected subgraphs that can be detected using graph theoretic approaches [2]. However, their detection efficacy in practice is unclear: a recent study on the Chinese Renren network casts doubt on their assumption, which reveals that the majority of Sybils are actively friending real users, and successfully integrating themselves into the social graph [3].

In this paper, we describe a new approach to Sybil detection rooted in the fundamental behavioral patterns that separate real and Sybil users. Specifically, we propose the use of server-side *clickstreams*, which are traces of click-through events generated by online users during each web browsing "session." Intuitively, Sybils and real users have very different goals in their usage of social networks: where real users likely partake of numerous features in the system, Sybils focus on specific actions (*i.e.* acquiring friends and disseminating spam) while trying to maximize utility per time spent. We hypothesize that these differences will manifest as significantly different (and distinctive) patterns in clickstreams, making them effective tools for "profiling" user behavior.

Our work focuses on building a practical clickstream model for accurate Sybil detection. We develop models that encode distinct click event sequences and inter-event gaps in clickstreams. We build weighted graphs of these sequences that capture pairwise "similarity distance" between clickstreams, and apply clustering to identify groups of user behavior patterns. By identifying suspicious behavioral clusters that deviate from normal users, we can detect Sybil accounts in social networks.

II. System Overview

At a high-level, we design our Sybil detection system based on a key observation: Sybils and real users have different goals in using social networks, thus exhibit different browsing patterns reflected in their clickstreams (*e.g.* what pages they click on, click speed, the ordering of clicks). We aim to build models to effectively cluster user clickstreams that exhibit similar behaviors. With well-clustered clickstreams, we can identify Sybils whose clickstream clusters deviate from normal users.

A. Clickstream Clustering

Before describing how we perform clustering, we first need to define user clickstreams. A clickstream is the sequence of HTTP requests made by a user to the social network. Thus each request (or click) is characterized by a timestamp, an userID, and a *click event*. The click event is derived from the request URL, and describes the action the user is undertaking. So a user's clickstream is represented as an in-order sequence of clicks along with inter-event gaps between clicks. For example: $a(t_1)c(t_2)a(t_3)d(t_4)b$ where a, b, c, d are click events, and t_i is the time interval between the i^{th} and $(i+1)^{th}$ event.

To effectively cluster similar clickstreams, we map clickstreams to a *similarity graph*, where clickstreams (vertices) are connected using weighted edges that capture pairwise similarity. We apply graph partitioning (using METIS[1]) to identify clusters that represent specific behavior (click) patterns. Ideally, Sybil and real users will fall in different behavior clusters and these clusters are the basis for our Sybil detection.

Clickstream Similarity. The key of our approach is how to map user's clickstreams to this similarity graph. Put in another way, how to compute the "distance" between two clickstreams so that we can best separate Sybil with normal user's clickstreams in the graph.

Through a detailed evaluation [4], we find the common subsequences method (with counts) beats other alternatives.

[1]http://glaros.dtc.umn.edu/gkhome/views/metis

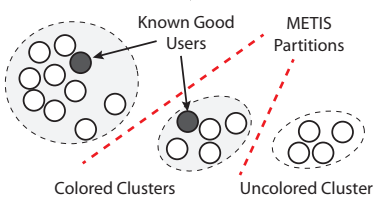


Fig. 1. Unsupervised clustering with coloring.

It works as follows: for a clickstream sequence[2] S_1, S_2 and a chosen k, we first compute the set of all possible subsequences from both sequences as $T = T_k(S_1) \cup T_k(S_2)$. Next, we count the frequency of each subsequence within each sequence i ($i = 1, 2$) as array $[c_{i1}, c_{i2}, ..., c_{in}]$ where $n = |T|$. Finally, the distance between S_1 and S_2 can be computed as the normalized *Euclidean Distance* between the two arrays: $D(S_1, S_2) = \frac{1}{\sqrt{2}}\sqrt{\sum_{j=1}^{n}(c_{1j} - c_{2j})^2}$. Note that this distance function inherently incorporates clickstream features like types of click events, click transitions and click speed.

So in the end of clickstream clustering, we can produce a clustered sequence similarity graph, where Sybil clusters (more Sybil sequences than normal in the cluster) and normal user clusters (more normal sequences than Sybils in the cluster) are well distinguished.

B. Sybil Detection

To classify a new clickstream given an existing clustered sequence similarity graph, we must determine how to "re-cluster" new sequences into the existing graph. This is essentially a Sybil detection process.

Incremental Detection. For a given unclassified clickstream sequence, we compute its average distance to all existing sequences in each cluster. The unclassified sequence is then added to the cluster with the closest average distance. The new sequence is classified as Sybil if it is added to a Sybil cluster (more Sybil clickstreams than normal) and vice versa.

In practice, we can make this detection process less computational-intensive, by precomputing a "center" for each cluster. In this way, when an unclassified clickstream comes in, we don't have to compute its distance to all clickstreams in the graph, but only need to compute its distances to cluster centers. This will help our system scale to large social networks.

Supervised vs. Unsupervised. As described, the system requires knowing whether a specific cluster in the similarity graph is a Sybil cluster or normal cluster, before running the incremental detection. This lead to a shortcoming: it must be trained using large samples of ground-truth data. In practice, this *supervised* approach will require social network administrators to put efforts to collect large ground-truth samples, which is a burdensome task.

Here, we also develop an *unsupervised* Sybil detection approach that requires only a small, constant amount of ground-truth. The key idea is to build a clustered sequence similarity graph as before. But instead of using full ground-truth of all clickstreams to mark a cluster as Sybil or normal, we only need a small number of clickstreams of known real users as "seeds" that color the clusters they reside in (Figure 1). We *color* all clusters that include a *seed* sequence as "normal," while uncolored clusters are assumed to be "Sybil." Since normal users are likely to fall under a small number of behavioral profiles (clusters in the graph), we expect a small fixed number of seeds will be sufficient to color all clusters of normal user clickstreams. Once the system is trained in this manner, it can be used incrementally to detect more Sybils over time.

[2]We discretize (bucketize) inter-arrival times so that we can encode them into the clickstream sequence.

III. System Evaluation

Ground-truth Datasets. We evaluate our system using a large ground-truth dataset obtained from Renren, the largest social network in China with 220 million users. The security team of Renren provided us a ground-truth dataset (anonymized) with 9994 Sybils and 5998 normal users. The dataset contains users' complete clickstream history during March and April 2011. In total, our dataset includes 1,008,031 and 5,856,941 clicks for Sybils and normal users respectively.

Detection Performance. We run our system on the ground-truth dataset to test its efficacy. To test the supervised detector, we first split the ground-truth dataset equally into two parts (at random) as training and testing datasets. Then, we build similarity graph of clickstreams using training dataset and produce initial behavior clusters (labeled by ground-truth). Finally, we incrementally add new testing clickstreams to existing clusters to simulate the Sybil detection process. Our system produces <1% false positives (classifying normal as Sybil) and <3% false negatives (classifying Sybil as normal), demonstrating its high accuracy.

To investigate the unsupervised detection approach, we perform a similar experiment, but assuming there is no ground-truth information available for training. We select x number of known-good users mixed in the training data as seed to color the behavior clusters. In this case, we find a handful of seeds (*e.g.* x=400, 7% of total 6000 training samples) are enough to accurately color normal clusters. The unsupervised detector produces 4% false negatives, but still remains <1% false positives. This indicates that our system can perform efficiently with minor ground-truth from social networks.

IV. Real-world Deployment

With the help of supportive collaborators at both Renren and LinkedIn, we were able to ship prototype code to their security teams for internal testing on fresh data. The initial results are very positive. We plan to continue working with them to improve the system and implement it in production.

References

[1] H. Gao, J. Hu, C. Wilson, Z. Li, Y. Chen, and B. Y. Zhao, "Detecting and characterizing social spam campaigns," in *Proc. of IMC*, 2010.

[2] H. Yu *et al.*, "Sybilguard: defending against sybil attacks via social networks," in *Proc. of SIGCOMM*, 2006.

[3] Z. Yang, C. Wilson, X. Wang, T. Gao, B. Y. Zhao, and Y. Dai, "Uncovering social network sybils in the wild," in *Proc. of IMC*, 2011.

[4] G. Wang, T. Konolige, C. Wilson, X. Wang, H. Zheng, and B. Y. Zhao, "You are how you click: Clickstream analysis for sybil detection," in *Proc of USENIX Security*, 2013.

Augmented Reality-based Remote Collaboration

Steffen Gauglitz Matthew Turk Tobias Höllerer
Four Eyes Lab, Department of Computer Science, University of California, Santa Barbara
{sgauglitz, mturk, holl}@cs.ucsb.edu

***Abstract*—We describe a framework to support mobile remote collaboration on tasks that involve the physical environment. By building on state-of-the-art computer vision and augmented reality techniques, we enable more immersive and more direct interaction with the remote environment than what is possible with today's ubiquitous video conferencing paradigm. In addition to the general framework, we describe a prototype implementation and user study, which demonstrate significant benefits of the proposed paradigm.**

I. Introduction

With the widespread deployment of fast data connections and the availability of a variety of sensors for different modalities, the potential of remote collaboration has greatly increased. While the now ubiquitous video conferencing applications take advantage of some of this potential, the use of video between remote and local users is limited to passively watching disjoint video feeds, leaving much to be desired regarding direct interaction with the remote physical environment. This is aptly summarized by Gelb et al. [4]: "Current systems, however, do a poor job of integrating video streams ... Real and virtual content are unnaturally separated, leading to problems with nonverbal communication and the overall conference experience." For example, gesturing and pointing are very natural and effective aspects of human communication, without which communication can be ineffective ("If I could just point to it, its right there" [1]). Thus, teleconference-like applications have been largely successful when the matter at hand can be discussed verbally or with the help of purely digital data (such as presentation slides), but they hit severe limitations when real-world objects or environments are involved.

We suggest that mobile augmented reality (AR) provides a natural and user-friendly way to extend the video conferencing paradigm. The conceptual framework for this paradigm is depicted in Fig. 1: Using the live video stream from a camera in the local user's environment (such as a webcam or mobile smartphone camera) and computer vision-based tracking and mapping techniques, the system creates a model of the environment. The remote user is able to browse this model (that is, control the viewpoint of a virtual camera in this virtual environment) and communicate spatial information to the local user by creating virtual annotations in it (in addition to all communication means supported by today's video conferencing paradigm).

Our paradigm is compatible with off-the-shelf hardware systems that are already ubiquitous (e.g., current generation smart phones) but scales to more advanced high-end systems as well. The system allows the user to be completely mobile and does not require preparation of or information about the environment of any kind. We believe that the applicability of telecollaboration systems will be increased substantially by allowing for visual and spatial interaction.

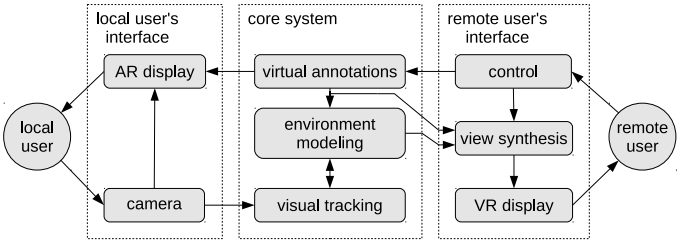


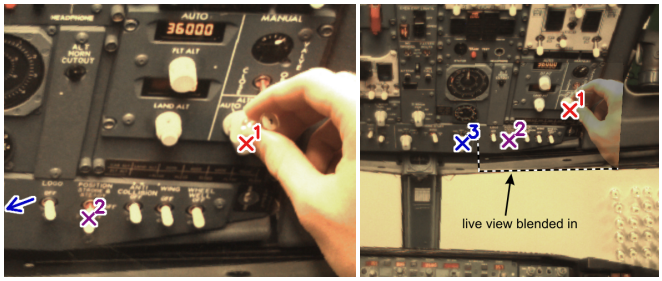
Fig. 1. Conceptual overview of the proposed framework for unobtrusive, mobile telecollaboration that includes the physical environment [2].

II. Prototype Implementation and User Study

We designed and implemented a first prototype instantiation of the framework in Fig. 1 and conducted a formal user study, comparing it to traditional interfaces. This first prototype does not exhaust the potential of the framework; it is less immersive and interactive than the vision described above. Nevertheless, we were able to demonstrate significant benefits.

For this prototype, the local user was holding a tablet (screen with camera mounted on back). The remote user used a standard PC and received the live camera stream from the local user. The scenario for our user study was that the remote user helped the local user to "operate" a mock-up airplane by pointing out the controls that had to be toggled. We provided three different interfaces: With **interface A**, the remote user saw the live video, but did not have any means of providing visual/spatial feedback. This is similar to using a current tablet PC with rear-facing camera and standard video conferencing software. With **interface B**, the remote user was able to click into the video view to create a marker that was visible on both screens. However, the marker was static with respect to image coordinates, so it appears to "swim away" from its original position if the tablet's camera is moved. **Interface C** implements the paradigm of Fig. 1. Here, the computer tracks the camera position and builds a model of the environment in the background. The markers that the remote user sets are anchored to this model; thus, they "stick" to their original positions despite movement of the camera. Additionally, the remote user had some control over his viewpoint, since the model of the environment allows to synthesize live views of the scene to some degree. The two camera views — the local user's live camera and the remote user's virtual camera — remain registered to each other and all data (live images from the local user and markers set by the remote user) are transferred and displayed immediately, as shown in Fig. 2.

The main results of the study were as follows: First,



(a) local user's view of the scene (b) remote user's view of the scene

Fig. 2. Example view of the task environment for the local and the remote user. The remote user's viewpoint is different from that of the local user, but the view is *live*: the live video stream is — correctly registered with the new viewpoint — blended in. The remote user has control over the colored markers, which are displayed in both views, correctly registered to the respective viewpoints (even if outside the current field of view, as is the case for the blue marker on the left).

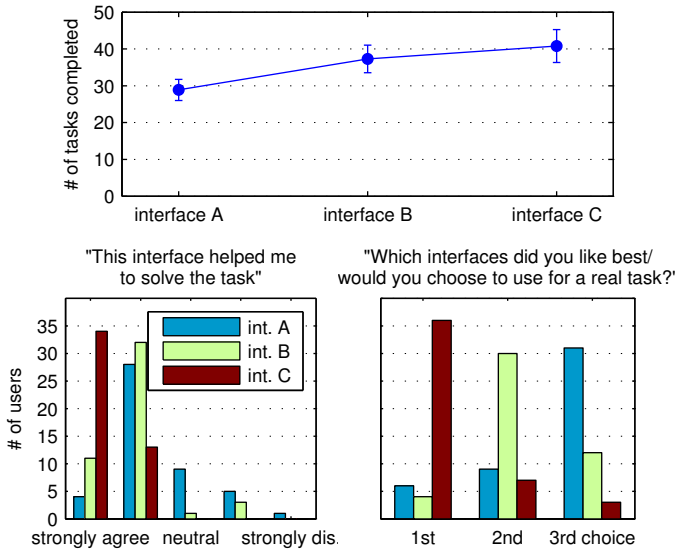


Fig. 3. Main results of the user study. Top: number of tasks completed (mean and 95% confidence intervals) as a function of the three different interfaces. Bottom: User responses from post-study questionnaire (selected results).

users completed significantly more tasks (in the same amount of time) with interface C compared to A (Fig. 3 top). (No significant difference was found between B and C.) Second, 79% of the users preferred interface C and would use it for a real task (Fig. 3 bottom). Details on the statistical analysis, further results and in-depth discussion can be found in [2]. It is important to note that these results were obtained despite the fact that interfaces A and B were failsafe, while interface C had to be built upon non-perfect visual tracking, which led to occasional tracking loss and thus loss of task time, which was not compensated for. That is, as the technology improves, the demonstrated benefits can be expected to increase. We take these results as a strong indication that the proposed framework is promising.

III. On-going Work

As stated above, the described first prototype does not exhaust the potential of the proposed framework. Several components may be substituted with more flexible, more powerful, or more immersive components. Additionally, our prototype has one particular technical limitation (described in the next paragraph). In this section, we briefly mention some of our on-going work that aims at addressing those limitations and increasing the level of immersion.

3D environment modeling. The prototype described in Section II can operate only in planar environments due to the kind of tracking and modeling system it uses (namely, it operates with *homographies*). While tracking and modeling systems for general 3D environments — commonly referred to as *simultaneous localization and mapping* (SLAM) — exist, these systems require *parallax-inducing* camera motion: they fail if the user stands still or rotates on the spot [3]. We had thus consciously decided to not use a SLAM system for the first prototype. Since then, we designed a concept to real-time tracking and mapping which explicitly supports both parallax-inducing and rotation-only camera motion. Approach and results are described in detail in [3].

Viewpoint control (virtual navigation). Thus far, the flexibility which the remote user has in choosing his/her viewpoint is very limited. Ideally, we would like him/her to be able to *freely explore* the remote environment (by means of the synthesized model as a proxy). Virtual navigation is well-known to be challenging, especially if high-dimensional spaces (free viewpoint control in 3D has at least six degrees of freedom) are mapped to lower dimensional inputs (e.g., two degrees of freedom of a mouse or touchscreen). Consequently, it is a large and active research area. However, we face an additional challenge: Most related work assumes that the virtual environment is *known completely* (e.g., the virtual world in a game) and can thus be rendered from any viewpoint. Here, however, the environment is known only as far as it has been observed the local user's camera and thus can be rendered only from a restricted set of viewpoints. The virtual camera navigation has to take this into account and guide the remote user to *useful* views of the scene. To our knowledge, this is a novel research area to be explored.

Further areas to be explored include support for more complex virtual annotations (such as drawings, or direct overlay of hand gestures), and use of more immersive display systems (such as projector-based systems, which project their image directly into the world). For many of these aspects, related work exists, but has specific shortcomings (such as requirements of prepared environments or static cameras), which can be alleviated by our framework. All of these aspects deserve further investigation and bear the potential to further increase the applicability of mobile interactive telecollaboration interfaces.

References

[1] S. R. Fussell, L. D. Setlock, J. Yang, J. Ou, E. Mauer, and A. D. I. Kramer, "Gestures over video streams to support remote collaboration on physical tasks," *Hum.-Comput. Interact.*, 19:273–309, 2004.

[2] S. Gauglitz, C. Lee, M. Turk, and T. Höllerer, "Integrating the physical environment into mobile remote collaboration," *Proc. MobileHCI* 2012.

[3] S. Gauglitz, C. Sweeney, J. Ventura, M. Turk, and T. Höllerer, "Live tracking and mapping from both general and rotation-only camera motion," *Proc. ISMAR* 2012.

[4] D. Gelb, A. Subramanian, and K.-H. Tan, "Augmented reality for immersive remote collaboration," *Workshop on Person-Oriented Vision* 2011.

A Brief Overview of Hairball: Lint-inspired Static Analysis of Scratch Projects

Bryce Boe, Charlotte Hill, Phillip Conrad, Diana Franklin
Department of Computer Science, UCSB
[bboe, charlottehill, pconrad, franklin]@cs.ucsb.edu

***Abstract*—Scratch programming has risen in prominence, not only as a potential language for K-12 computer science, but also in introductory college courses. Unfortunately, grading Scratch programs is time-consuming, requiring manual execution of each program. Automation of this process is greatly complicated by the very reason Scratch is an attractive introductory language—the projects are multimedia in nature, requiring eyes and ears to fully appreciate.**

We developed Hairball, an automated system that can be used both by a student to point out potential errors or unsafe practices, and by a grader to assist in inspecting the implementation of Scratch programs. Hairball focuses on the implementation, including safe/robust programming practices, providing a "lint-like" tool for Scratch.

Our summary evaluation shows that Hairball is very useful in conjunction with manual analysis. Overall, Hairball was actually slightly more accurate than manual analysis at labeling instances of computer science concepts in Scratch programs. Our results suggest if Hairball was only used to identify correctly implemented instances, with manual analysis used on the remainder, it would remove 76% of the instances for the manual analysis and assist in the rest, with a false positive rate of less than 0.5%.

I. Introduction

There is a movement toward less industrial and more engaging projects and languages for introductory and AP computer science courses. This movement includes the push for Python with Multimedia approaches [3], [8], the various approaches to the AP CS Principles course [9], as well as Alice [2] and Scratch [7].

One drawback of visual and auditory projects is that their evaluation can be more difficult than traditional text-based programming assignments. A common and straightforward practice in evaluating text-based assignments is to perform functional testing. That is, to write a script to run all submitted programs and compare their output with solution files [5]. When students are given creative freedom with a sensory project, there is not a text-based output file with which to diff. With Scratch, for example, evaluation typically requires that each project be individually opened and run. Inspection of the code requires many mouse clicks in order to traverse through many levels of the interface.

To assist with assessment of Scratch projects, we developed a static analysis tool. Inspired by the Scratch mascot (a cat), and the concept of lint (a static analysis utility for C that looks for potential defects [6]), we call our system Hairball. Hairball can quickly differentiate between Scratch programs that do, or do not, contain certain targeted constructs, and is particularly helpful for identifying instances of particular constructs and implementations that are not robust but may not immediately cause obvious errors at runtime. Manual analysis, however, is still needed to evaluate the overall aesthetic effect and cohesion of a visual or auditory project.

In the remainder of this brief paper, we will only describe the overall results from running Hairball on 53 projects covering four CS concepts. For the complete results and additional discussion please see our full paper, "Hairball: Lint-inspired Static Analysis of Scratch Projects" published in SIGCSE 2013. [1].

II. Methodology

We tested Hairball on 58 projects submitted during a two week summer camp for middle school students hosted at UCSB in the summer of 2012. The purpose of the camp was to both build interest in computer science, and discover whether or not students would learn the computer science concepts in a "fun-oriented" summer camp. There were five assignments total, the first four each introducing a new computer science concept, and the fifth utilizing all four concepts [4].

We first performed a manual analysis on all 58 of the submitted Scratch projects. Three members of our project staff independently analyzed the first five projects submitted for a given assignment using a common rubric. We discussed any discrepancies in our scores and after coming to agreement, we analyzed the remaining projects. Once again, any score discrepancies were reconciled.

Hairball was programmed to match the methodology agreed upon by the staff members and then run on all 58 projects. When there were discrepancies between Hairball and the manual analysis, there was a second manual analysis to determine which was correct: Hairball or the manual analysis.

III. Results

In this section, we present the aggregate results of using Hairball to assist in determining the level of competence demonstrated by students' Scratch projects for four Scratch concepts. The four Scratch concepts are state initialization, broadcast and receive, say and sound block synchronization, and complex animation. For each concept, we compare the labels Hairball assigned to instances of the concept with those assigned via manual analysis. We will use the second manual analysis as the ground truth, and look at both false positive

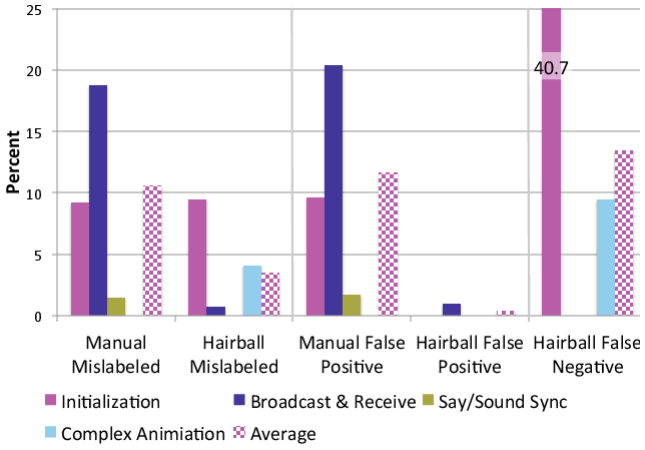


Fig. 1. Provides a summary of the percent of mislabeled, false positive, and false negative instances from manual analysis and Hairball for each of the four CS concepts and the average. The "Manual False Negative" category was omitted as manual analysis resulted in zero false negatives. The y-axis is truncated for the smaller values, thus the tallest bar should extend to 40.7%. Missing bars represent 0%.

and false negative rates for Hairball and the manual analysis. We consider a false positive to be an instance that was labeled correct, when in fact it is not, and a false negative to be an instance that is actually correct, but was not labeled as such. For manual analysis both false positives and false negatives represent the inaccuracies of manual assessment. For Hairball, false negatives can be considered warnings, i.e., they are used to indicate the need for additional manual analysis. However, any false positives produced by Hairball are cause for concern.

1) Summary Results: Figure 1 shows three sets of results across all four CS concepts and the overall average. The first is the mislabel rate of manual analysis and Hairball. Percentages closer to zero indicate higher accuracy. We can see that Hairball is actually slightly more accurate overall than manual analysis, largely in part of its accuracy in labeling broadcast and receive instances.

The second set of results are the rate of false positives. Manual analysis, with an overall false positive rate of 11.7%, indicates that manual analysis is quite error prone. On the other hand, Hairball's false positive rate of 0.4% strongly indicates that instances Hairball labels as "correct" can be trusted.

Finally, the third set of results are the rate of false negatives. The lack of false negatives for manual analysis makes sense, considering Hairball was created according to the first manual analysis. Although Hairball has an overall false negative rate of 13.5%, we believe this rate to be acceptable due to the fact that four out of five instances in our ground truth set were labeled "correct".

IV. Conclusions

We presented a segment of our study showing a new static analysis tool, Hairball, that provides an extendable framework for automatically analyzing Scratch programs. In addition, we provide an initial set of plugins that analyze the implementation of Scratch programs for competence in four areas: initialization, broadcast and receive, say and sound synchronization, and animation. Our evaluation shows that Hairball is extremely usful in identifying correctly implemented instances, with a false positive rate of less than 0.5%. Overall, the mislabel rate of Hairball is less than half that of manual analysis. Therefore, we propose Hairball as an addition to, not replacement of, manual analysis.

References

[1] B. Boe, C. Hill, M. Len, G. Dreschler, P. Conrad, and D. Franklin. Hairball: Lint-inspired static analysis of scratch projects. In *SIGCSE '13*, March 2013.

[2] S. Cooper, W. Dann, and R. Pausch. Teaching objects-first in introductory computer science. In *Proceedings of the 34th SIGCSE technical symposium on Computer science education*, SIGCSE '03, pages 191–195. ACM, 2003.

[3] A. Forte and M. Guzdial. Computers for communication, not calculation: Media as a motivation and context for learning. In *Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 4 - Volume 4*, HICSS '04, pages 40096.1–, Washington, DC, USA, 2004. IEEE Computer Society.

[4] D. Franklin, P. Conrad, B. Boe, K. Nilsen, C. Hill, M. Len, G. Dreschler, and G. Aldana. Assessment of computer science learning in a scratch-based outreach program. In *Proceedings of the 44th SIGCSE technical symposium on Computer science education*, SIGCSE '13. ACM, 2013.

[5] D. Jackson and M. Usher. Grading student programs using assyst. In *Proceedings of the twenty-eighth SIGCSE technical symposium on Computer science education*, SIGCSE '97, pages 335–339. ACM, 1997.

[6] S. C. Johnson. Lint, a c program checker. In *COMP. SCI. TECH. REP*, pages 78–1273, 1978.

[7] J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond. The scratch programming language and environment. *Trans. Comput. Educ.*, 10(4):16:1–16:15, Nov. 2010.

[8] B. Simon, P. Kinnunen, L. Porter, and D. Zazkis. Experience report: Cs1 for majors with media computation. In *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education*, ITiCSE '10, pages 214–218. ACM, 2010.

[9] L. Snyder, T. Barnes, D. Garcia, J. Paul, and B. Simon. The first five computer science principles pilots: summary and comparisons. *ACM Inroads*, 3(2):54–57, June 2012.

The Design of notJS: an Intermediate Representation of JavaScript for Abstract Interpretation

Vineeth Kashyap Kyle Dewey Ethan A. Kuefner Ben Hardekopf
University Of California, Santa Barbara
{vineeth, kyledewey, eakuefner, benh}@cs.ucsb.edu

***Abstract*—JavaScript has become pervasive. Static analysis tools for JavaScript can help construct secure, correct and fast JavaScript code. We built JSAI, a novel design for static analysis of JavaScript through abstract interpretation. JSAI is designed to be sound, extensible and practical. In this work, we describe the design of a core part of the JSAI toolchain: the intermediate representation (IR) of JavaScript called notJS that is used for abstract interpretation. The use of such an IR simplifies the construction of the abstract interpreter, and makes the abstract interpreter precise and efficient. We have formally specified the translation of JavaScript into notJS, and implemented it in Scala.**

I. Introduction

JavaScript is an epidemic. Web developers use JavaScript to enrich user experience via dynamic content ranging from scripts to enhance a web page's appearance, to full-blown web applications, to extending the functionality of web browsers in the form of browser addons. Desktop developers use JavaScript, e.g., for OS applications in Windows 8. A growing list of compilers use JavaScript as their target language. JavaScript's growing prominence means that secure, correct, and fast JavaScript code is becoming ever more critical. Static analysis traditionally plays a large role in providing these characteristics: it can be used for security auditing, error-checking, debugging, optimization, and program refactoring, among other uses. Thus, a sound, precise static analysis platform for JavaScript can be of enormous advantage.

JavaScript itself is a dynamic language in which almost everything is dynamically modifiable: it is dynamically typed, object properties (the JavaScript name for object members) can be dynamically inserted and deleted, inheritance via prototype chains can be changed dynamically, and more. This dynamism makes static analysis of JavaScript a significant challenge.

The current state of the art for static analysis of JavaScript takes one of two approaches: either **(1)** an unsound dataflow analysis-based approach using baked-in data abstractions and baked-in context- and heap-sensitivities, or **(2)** a formally-specified type system, proven sound with respect to a specific JavaScript formal semantics but restricted to a small subset of the full JavaScript language, often requiring the added burden of annotations.

We have built JSAI (the JavaScript Abstract Interpreter), which is a fully **automatic** (no annotation burden), **sound** (no false negatives), **extensible** (pluggable abstract data domains and variety of context- and heap-sensitivities) and **practical** (can scale to real-world JavaScript programs of non-trivial size) static analysis tool for JavaScript. In the rest of the paper, we discuss a core part of the JSAI toolchain: the IR representation of JavaScript called notJS. The design of notJS is critical for the construction of JSAI—the IR dictates the performance and precision of the abstract interpreter, and vastly simplifies the formalisms and implementation. We have experimented with several iterations of notJS, and in the next section we provide insights into the key aspects of our current design. We have formally specified a translation from JavaScript to notJS, and given notJS a concrete and abstract semantics (the formal specifications and their implementations in Scala are out of scope for this paper for the lack of space).

II. Designing the notJS Intermediate Language

In order to claim that JSAI is sound, we should be able to show that our abstract semantics soundly over approximates the concrete semantics of JavaScript. However, JavaScript only has informal english-specified semantics, which is not amenable to any soundness proofs. Thus, the first logical step of constructing JSAI was to provide a concrete semantics to JavaScript, such that the concrete semantics allows simple, efficient and precise abstraction.

With this step, the choice we faced was whether to **(1)** provide semantic formalisms directly for the JavaScript language itself; or **(2)** translate JavaScript into an intermediate representation and provide semantic formalisms for that. Several previous works have provided semantics directly for the JavaScript language [1], adhering as closely as possible to the ECMA standard. The lesson we learn from those efforts is that the resulting formalisms are large, complex, and difficult to reason about; abstracting those semantics would also be difficult and inefficient. Therefore, we chose the other option by designing an intermediate language called notJS, along with a formally-specified translation from JavaScript to notJS. The semantics of notJS is much smaller and simpler than the other approach, and hence easier to formalize and implement. Other JavaScript formalizations have used a similar strategy, e.g., Guha et al's λ_{JS} [2]; our approach is similar in concept, but our IR design is different. The main difference is the intended goals: λ_{JS} was designed for proving soundness of type systems (and thus they provide a minimal core language which results in tremendous code bloat), while notJS was designed for abstract interpretation.

Figure 1 shows the abstract syntax of notJS, which was carefully designed with the ultimate goal of making abstract interpretation simple, precise, and efficient. Notably absent from the syntax are JavaScript's builtin methods (e.g., eval) and objects (e.g,. Date). These are actually members of the global object (constructed prior to a program's execution) rather than part of the language syntax. The notJS IR has

$$n \in Num \qquad b \in Bool \qquad str \in String \qquad x \in Variable \qquad \ell \in Label$$

$$\begin{aligned}
s \in Stmt ::= {} & \vec{s_i} \mid \textbf{if } e\ s_1\ s_2 \mid \textbf{while } e\ s \mid x := e \mid x := e_1(e_2, e_3) \mid x := \textbf{new } e_1(e_2) \\
& \mid x := \textbf{newfun } m\ n \mid x := \textbf{toobj } e \mid x := \textbf{del } e_1.e_2 \mid e_1.e_2 := e_3 \\
& \mid \textbf{throw } e \mid \textbf{try-catch-fin } s_1\ x\ s_2\ s_3 \mid \ell\ s \mid \textbf{jump } \ell\ e \mid \textbf{for } x\ e\ s \\
e \in Exp ::= {} & n \mid b \mid str \mid \textbf{undef} \mid \textbf{null} \mid x \mid m \mid e_1 \oplus e_2 \mid \odot e \\
d \in Decl ::= {} & \textbf{decl } \overrightarrow{x_i = e_i} \textbf{ in } s \\
m \in Method ::= {} & (\texttt{self}, \texttt{args}) \Rightarrow d \mid (\texttt{self}, \texttt{args}) \Rightarrow s \\
\oplus \in BinaryOp ::= {} & + \mid - \mid \times \mid \div \mid \% \mid \ll \mid \gg \mid \ggg \mid < \mid \leq \mid \ \& \ \mid \ '|' \ \mid \veebar \\
& \mid \textbf{and} \mid \textbf{or} \mid +\!\!+ \mid \prec \mid \preceq \mid \approx \mid \equiv \mid \ . \ \mid \textbf{instanceof} \mid \textbf{in} \\
\odot \in UnaryOp ::= {} & - \mid \sim \mid \neg \mid \textbf{typeof} \mid \textbf{isprim} \mid \textbf{tobool} \mid \textbf{tostr} \mid \textbf{tonum}
\end{aligned}$$

Fig. 1: The abstract syntax of notJS provides canonical constructs that simplify JavaScript's behavior. The vector notation represents (by abuse of notation) an ordered sequence of unspecified length n, where i ranges from 1 to n.

constructs to handle all of the complexity of JavaScript. In addition to standard constructs (like sequencing, conditionals, loops, assignment, try-catch-finally and throw, etc.,) one can create objects through **new**, and closures through **newfun**. The **jump** statement allows handling `return`, `break` and `continue` constructs in JavaScript. The structure of the syntax can be boiled down to four important design decisions:

1) We separate the language into pure expressions ($e \in Exp$) that are guaranteed to terminate without throwing an exception, and impure statements ($s \in Stmt$) that do not have these guarantees. This decision directly impacts the formal semantics and implementation of notJS (in particular, it reduces the number of semantic rules by a third compared to a version that does not separate between pure expressions and impure statements). This is the first IR for JavaScript we are aware of that makes this design choice—it is a more radical choice than might first be apparent, because JavaScript's implicit conversions make it difficult to enforce this separation. The IR was carefully designed to make this possible.
2) We make JavaScript's implicit conversions explicit in notJS with **toobj**, **tostr**, **tonum**, and **tobool**. JavaScript's implicit conversions are complex and difficult to reason about, involving multiple steps and alternatives depending on the current state of the program. Operators are also simplified: $+$ performs numeric addition and $+\!\!+$ performs string concatenation (unlike the JavaScript `+` operator which is overloaded to do both based on the context). By making these decisions explicit in notJS, we greatly simplify the semantics.
3) We lower many of the JavaScript language constructs into simpler and more regular forms (e.g., most of the JavaScript loop constructs are lowered to **while** loops). However, we have left some JavaScript constructs unlowered. The prime example of this is JavaScript's `for..in` loop, which we leave mostly intact as **for** $x\ e\ s$. Leaving this construct unlowered gives the abstract interpreter the chance to treat it specially, thereby potentially increasing precision and performance.
4) We have simplified the behavior for method calls. JavaScript methods have unintuitive rules for the implicit `this` parameter (which often, but not always, provides the address of a method's receiver object). We simplify this reasoning by making the formerly implicit parameter explicit as `self`. JavaScript also provides a form of reflection using the `arguments` array, which aliases the method's actual parameters. Method callers can provide fewer or more arguments than there are parameters; any extra arguments are also stored in the `arguments` array. In the abstract syntax, methods always take exactly two arguments: `self` and `args`, and notJS always passes all of its arguments via `args`, which emulates the `arguments` array.

We specify the notJS concrete semantics (and abstract semantics) as an abstract machine-based smallstep semantics. One can think of this semantics as a state transition system, wherein we formally define a notion of *state* and a set of *transition rules* that connect states. We omit further details of the concrete semantics (and the abstract semantics) for lack of space. We tested the translation from JavaScript to notJS, semantics, and our Scala implementation thereof by comparing its behavior with that of an industry standard JavaScript engine, SpiderMonkey by Mozilla. We constructed a test suite of 243 JavaScript programs, some hand-crafted to exercise various parts of the semantics and some taken from existing JavaScript programs used to test commercial JavaScript implementations. We then ran all of the tests on SpiderMonkey and on our concrete interpreter, and we verified that they produce identical output. While we can never completely guarantee that the notJS semantics matches the ECMA specification, we can do as well as any JavaScript implementation, which goes through the same sort of testing process.

REFERENCES

[1] S. Maffeis, J. C. Mitchell, and A. Taly, "An operational semantics for javascript," in *Asian Symposium on Programming Languages and Systems*, 2008.

[2] A. Guha, C. Saftoiu, and S. Krishnamurthi, "The essence of javascript," in *European conference on Object-oriented programming*, 2010.

Dynamic Machine-Code Generation for Quantum Rotations

Daniel Kudrow, Kenneth Bier, Zhaoxia Deng, Diana Franklin, Frederic T. Chong
University of California, Santa Barbara
{dkudrow, kbier, zhaoxia, franklin, chong}@cs.ucsb.edu

***Abstract*—A critical issue in quantum computer architecture is the implementation of rotation gates. Quantum algorithms require arbitrary rotation angles, while quantum technologies and error correction codes provide only for discrete angles and operators. A sequence of quantum machine instructions must be generated to approximate the arbitrary rotation to the required precision. Previous work has been largely motivated by Shor's factorization algorithm which can be compiled statically. We examine a larger range of benchmarks and find that some applications require dynamic compilation of rotation gates.**

We introduce a new method for compiling arbitrary rotations dynamically, designed to minimize compilation time. The new method reduces compilation time by up to five orders of magnitude while increasing code size by one order of magnitude when compared to existing techniques.

I. Introduction

Previous work in quantum computer architecture has largely used Shor's algorithm as a driving application[4]. Due to the relative simplicity of Shor's algorithm, these studies have focused on error correction and communication in the context of quantum addition. Recent efforts in the quantum computing community have sought to define a much broader set of quantum benchmark algorithms and to create a tool chain to compile quantum programs and synthesize quantum architectures. We examined these benchmarks and found that an important challenge lies in generating sequences of quantum operators (quantum assembly) to implement a "quantum rotation" (also known as a quantum phase shift operator) at the algorithm level.

This paper introduces a new technique for compiling arbitrary rotations aimed at facilitating dynamic compilation. Compilation time is reduced by five orders of magnitude at a cost of 10X larger executable.

A. Background

The fundamental unit of quantum information is the quantum bit or *qubit*. Whereas a classical bit is confined to existing in either a high or a low state, a qubit can exist in both states simultaneously. This phenomenon is known as *superposition* and a qubit exists in superposition until it is measured. Upon measurement a qubit "collapses" into a classical bit, taking and keeping a value of either high or low. A qubit in superposition is described by the probability with which it will be found in each classical state once measured.

Qubits have another property called *phase*. While phase does not effect the outcome of a measurement on a single qubit, it is important in describing interactions between multiple qubits. The ability to manipulate the phase of a qubit is crucial to many tasks that involve the evolution of multi-qubit systems. The focus of this work is the *phase-shift gate* which is a quantum operation that changes the phase of a single qubit, often in preparation for an interaction with another qubit. The phase-shift gate is also known as the *rotation gate* because it can be visualized as a rotation about the z-axis of the Bloch Sphere. Rotation gates feature prominently in many quantum algorithms and are integral to quantum implementations of the Fourier Transform, phase estimation, and quantum random walks.

B. Arbitrary Rotations

Some rotation angles, such as the $\frac{\pi}{2}$ (Z gate) and the $\frac{\pi}{8}$ (T gate), are directly supported by most quantum computing architectures making compilation trivial. However, rotations of arbitrary angles pose a challenge to quantum computer architects because they are parameterized by the range $[0, 2\pi]$ and thus form a continuous set. Because only discrete sets of gates can be implemented fault-tolerantly, arbitrary rotations must be approximated by sequences of discrete gates [1]. The compilation of an arbitrary rotation is the process of generating a sequence of gates from a given set that, when executed, provide a good approximation of the original rotation.

II. Dynamic Compilation

In many quantum applications that utilize rotation gates (including Shor's algorithm) the angle of each rotation can be determined before execution using static analysis techniques. In such cases rotations can be compiled along with the rest of the quantum algorithm in a static compilation. However, we have found that there are situations in which a rotation angle is dependent on the outcome of a qubit measurement. As the outcome of a qubit measurement cannot be calculated statically, compilation of these rotations must be deferred until the quantum program executes. When performing dynamic compilation, minimizing compilation time is critical because it is absorbed into the execution time of the target application.

A. Existing Techniques

Existing methods of compiling arbitrary rotations have assumed a static compilation approach. As a result, compilation time has not been a major design constraint because static compilation time does not effect the running time of the executable. The two examples we consider in our analysis are the Solovay-Kitaev Algorithm[2] which is widely used in the quantum computing community and the recently released

Single Qubit Circuit Toolkit[3] which greatly imrpoves on the performance of the former. Both of these methods aim to provide precise approximations to arbitrary rotation gates in a minimal number of gates without special consideration for compilation time.

B. Library Construction

As an alternative to the aforementioned techniques we propose a new method for approximating rotations that focuses on minimizing compilation time. In this technique, a set of rotations is pre-compiled (using a static compilation technique such as Solovay-Kitaev) and stored as a library. Arbitrary rotations are then rapidly assembled by concatenating rotations from this library. We are essentially trading storage (and a less optimized result) for execution time. The library rotations follow the form,

$$\left\{ \frac{(n-1)\pi}{n^1}, \frac{(n-2)\pi}{n^1}, ..., \frac{\pi}{n^1}, ..., \frac{\pi}{n^k} \right\}. \tag{1}$$

An arbitrary angle can be built by concatenating the elements in the set using the construction,

$$\theta = \sum_{k=1}^{k=k_{max}} C_k \frac{\pi}{n^k}. \tag{2}$$

where C is the k^{th} digit in the base n representation of θ.

As an example consider a binary construction where $n = 2$. The library would consist of the gates.

$$\left\{ \frac{\pi}{2^1}, \frac{\pi}{2^2}, \frac{\pi}{2^3}, ..., \frac{\pi}{2^k} \right\}. \tag{3}$$

Now constructing an arbitrary rotation is as simple as rewriting it as a binary fraction. For instance,

$$\theta = \frac{7\pi}{9} = 0.11\overline{000111}. \tag{4}$$

By Adding the corresponding rotations in our library set,

$$\theta \approx \frac{\pi}{2^1} + \frac{\pi}{2^2} + \frac{\pi}{2^6} + \frac{\pi}{2^7} + \frac{\pi}{2^8} + ... \tag{5}$$

we are guaranteed an approximation to within $\frac{\pi}{2^k}$ of the original angle.

This technique allows for faster compilation because there is no need to generate new sequences. Determining which library sequences to concatenate takes $|\log_k(\epsilon)|(n-1)$ iterations to achieve an accuracy of ϵ or better. However, the cost for this speed comes in the form of increased sequence length for the approximation. Because each library rotation has the same sequence length as a statically compiled rotation, a rotation created in this manner has a worst case sequence length that is $|\log_k(\alpha)|$ times greater than its statically compiled counterpart. This drawback can be mitigated by increasing the size of the library. By using a large value for n we achieve higher precision at each iteration of the construction. A library built around a base of $n = 100$ can achieve an accuracy of $\frac{\pi}{100}$ in the first term of the sum whereas a library built around $n = 10$ will achieve this after only after the second term resulting in a sequence that is twice as long in the worst case. However the first library will be $\frac{(100-1)k}{(10-1)k} = 11$ times larger than the second.

This method is also in a position to take advantage of optimized static compilation techniques that are too slow to be used dynamically. Consider SQCT which sacrifices compilation time to generate length-optimal sequences. Although it may not be fast enough to generate code dynamically, we can still use it to create a library. A library composed of compilations that surpass the precision requirement of the target can compensate for the loss of precision incurred by dynamic code generation. In this way, optimized techniques can still be useful for dynamic code generation.

III. Results

We evaluated our library construction technique against the two existing methods by performing compilations across a wide range of input parameters. Each technique was evaluated on the accuracy of the resultant compilation, the compilation time, and the number of gates required to simulate the input rotation.

Our results showed that the library construction compiles arbitrary rotations up to 100,000X faster than either of the other two methods for a given accuracy. The cost for this speedup is a 10X increase in the number of gates required to synthesize the rotation. We can smooth the trade-off between compilation time and executable size by expanding into the spatial dimension. By increasing the number of pre-compiled rotations, we can achieve the same precision in fewer concatenations. With a 10X larger library we are able to reduce the executable size overhead from 10X to below 5X.

IV. Conclusion

We have identified that some quantum algorithms require dynamic compilation in order to execute successfully. Our library construction method marks the first attempt at tackling this challenge by greatly reducing the time needed to compile arbitrary rotations with the aim of minimizing effect on application runtime during dynamic compilation.

References

[1] I. Chuang & M. Nielsen *Quantum Computing and Quantum Information*, Cambridge University Press, Cambridge, 2000.

[2] C. M. Dawson and M. A. Nielsen, *The Solovay-Kitaev Algorithm*, 2005.

[3] V. Kliuchnikov, D. Maslov and M. Mosca, *Fast and efficient exact synthesis of single qubit unitaries generated by Clifford and T gates*, 2013.

[4] Peter W. Shor, *Algorithms for quantum computation: discrete logarithms and factoring*, In Proceedings 35th Annual Symposium on Foundations of Computer Science, pages 124-134, 1994.

Detecting Twitter Compromises

Gianluca Stringhini
University of California, Santa Barbara
gianluca@cs.ucsb.edu

Manuel Egele
Carnegie Mellon University
megele@cmu.edu

***Abstract*—Twitter is one of the main sites to get up-to-date news. Recent events have shown that attacks that compromise high-profile Twitter accounts and spread false news can have big consequences, even on the financial market. To stay reliable, and trustworthy these accounts need to be protected from such attacks. Our study shows that such protection becomes possible because most high-profile accounts have very consistent behavior over time. In this paper, we show that it is often possible to detect Twitter compromises by learning the typical sending behavior of Twitter accounts of news outlets.**

I. Introduction

In recent times, attackers started to compromise high-profile Twitter accounts, to spread fake news [2], [1]. Compromised accounts can also cause consequences that extend beyond the social network itself, and can even influence stock markets [3]. These events show how critical Twitter has become, and how important it is to ensure that Twitter accounts, especially the most popular ones, are adequately protected against compromises. When a Twitter account gets compromised, the messages that this accounts sends are likely to show anomalies, compared to the account's normal behavior. In this paper, we present a system that is able to detect these changes in behavior and flag them as possible compromises.

II. Detecting Compromises

In previous work [5] we combined anomaly detection with statistical modelling to detect changes in the behavior of Twitter accounts. The underlying assumption is that a compromised account shows a behavior that is different from the historically-observed behavior of the unaffected account. To detect large-scale campaigns that compromise and misuse multiple accounts simultaneously, our previous work also analyzed whether these changes in behavior manifest themselves across different accounts. While such techniques work well to detect large-scale efforts to compromise social networking accounts, the recent attacks on high-profile accounts are targeted in nature. Thus, to detect such attacks, we can no longer assume that multiple accounts are compromised and show similar diverging behavior. Detecting targeted attacks is more challenging than detecting campaigns, because legitimate changes in behavior in individual accounts are more likely to cause false alarms.

Behavioral models. In our previous work we introduced a system that models the sending behavior of Twitter users, and flags an anomaly if the user sends a message that violates such behavior [5]. To this end, we defined six features that model the characteristics of a tweet. We then build a behavioral model based on the tweets that an account sent in the past.

- *Time (hour of the day)*. This feature characterizes at what time of the day a tweet was sent.
- *Message Source*. Users post tweets using their favorite application. This could be the standard web interface, or a Twitter client. This feature represents the application that was used to send the tweet.
- *Links*. The account's behavior with respect to links is characterized in this feature. These characteristics include the frequency of tweets that contain links as well as the sites that are linked.
- *Language*. This feature characterizes the language that the tweet was authored in.
- *Hashtags*. This feature takes into account the hashtags used in the tweet.
- *Mentions*. If anybody in the tweet was mentioned, this feature keeps track of that.

In general, these features learn what is typical for a user, and flag an anomaly if a behavior that has never been seen before arises. For example, a message mentioning a contact that the user has never interacted with could be suspicious.

High-profile account characteristics. The recent targeted attacks were performed against high-profile media accounts. The behavior of these accounts over time is significantly different from the behavior of regular Twitter accounts. More precisely, the behavioral profiles of media company accounts are very narrow. For example, updates to such accounts are commonly posted during regular business hours. Furthermore, because status updates are limited to 140 characters, tweets are mostly used as teasers consisting of the headline and a link to a site with the full news story. This behavior is consistent across the vast majority of media accounts that we investigated.

Collecting Data. Compromised high-profile Twitter accounts are often quickly detected. However, recent events illustrated that quick is often not quick enough. It is suspected that (automated) high frequency traders are responsible for the drop in the US stock markets after the recent compromise of the Associated Press Twitter account. Once a compromise has been identified and reported to Twitter, the account undergoes a cleaning process during which the legitimate owner regains control over the account and all spurious information is deleted from the account's timeline. Because of this cleaning process, it is challenging to collect accurate information pertaining to such incidents. Note that once Tweets are deleted they are no longer publicly available. Therefore, we have been collecting 10% of all public tweets sent on Twitter resulting in over 4 billion messages.

III. Evaluation

In this section, we evaluate the behavior of the Twitter accounts of 20 high-profile news agencies over time. To this end, we collected the public timelines of these 20 accounts and performed the behavioral profile extraction and evaluation presented in [5] for the most recent 500 tweets in these timelines. Since we assume that the intention of media accounts is to disseminate original content, our analysis discards all retweets and conversations (i.e., replies) contained in a timeline. Our analysis has shown that such interaction frequently diverts from observed regular behavior and is only used in exceptional cases by official news agency Twitter accounts.

Consistent Behavior. In previous work we evaluated the probability that a random new tweet violates the behavioral profile of an account as 4% [5]. We consider an account to show consistent behavior if the probability of any of its messages violating the accounts behavioral profile is significantly less than these 4%. As Table I illustrates this is the case for the majority of the media accounts that we analyzed.

Twitter Account	#Violations (%)	Twitter Account	#Violations (%)
FoxNews	0 (0%)	washingtonpost	1 (0%)
CNN	0 (0%)	lemondefr	2 (0%)
foxnewspolitics	0 (0%)	Reuters	5 (1%)
AP	0 (0%)	BostonGlobe	11 (2%)
latimes	0 (0%)	BBCNews	16 (3%)
BloombergNews	0 (0%)	msnbc	17 (3%)
HuffingtonPost	0 (0%)	NBCNews	19 (4%)
BW	0 (0%)	el_pais	22 (4%)
abcnews	1 (0%)	DerSPIEGEL	63 (13%)
nytimes	1 (0%)	guardian	100 (20%)

Table I
Behavioral profile violations of news agency Twitter accounts within most recent 500 tweets.

Analysis. The only two media accounts whose behavior was not vastly consistent are the accounts of the German news outlet *Der Spiegel* and the British *Guardian*. Most profile violations in the *@derspiegel* account arose because this account is not only used to disseminate news but also to interact with other reporters. For example, advertising and communication during a journalistic convention held at the Spiegel headquarters are also included in that timeline. We suspect that the Guardian Twitter feed is managed by a set of different actors who have different preferences in terms of Twitter clients and slightly different editing styles. Our system is currently not able to characterize accounts with such multi-variant behavior patterns.

A. Case Studies

In this section, we analyze two attacks on high-profile media accounts in more detail. Our previous research found that accounts are frequently compromised by tricking the user into authorizing malicious Twitter applications. In contrast, the two attacks that we detail here share the characteristic that the attackers used the Twitter web interface to distribute fake news. One can only log into a Twitter account if the account name and password are known. Thus, we can conclude that the attackers in these cases had full access to the compromised accounts allowing them to authorize additional applications or change the accounts password. These actions require full access to the account and cannot be performed if the attacker only tricked the user to authorize a malicious application. A detailed report [4] from another victim of a similar attack supports this assumption by detailing the phishing emails used.

Associated Press. On April 23^{rd} 2013, the Twitter account of the Associated Press (@AP) was compromised [3]. The account was misused to distribute false information about president Obama being hurt by an explosion in the White House. Comparing the behavioral profile of the @AP account against this message resulted in significant differences among many features that our system evaluates. For example, the fake news was posted via the Twitter website, whereas the legitimate owners of the @AP account commonly use the SocialFlow application to send status updates. Furthermore, the fake tweet did not include any links to additional information, a practice that the @AP account follows very consistently. Only two features in our behavioral model did not signify a change of behavior. The time when the tweet was sent (i.e., 10:07UTC) and the language of the tweet itself. The authors of the @AP account as well as the attackers used the English language to author their content.

FoxNews Politics. On July 4^{th} 2011, the Twitter account of Fox News' politics (@foxnewspolitics) division got compromised [2]. The attackers used this opportunity to distribute the information that president Obama got assassinated. This tweet violated almost all the features used by our system. For example, the tweet was sent in the middle of the night (i.e., 23:24UTC), through the main Twitter web site. Furthermore, it did not include a link to the full story on the Fox News website. The tweet also made extensive use of hashtags and mentions, a practice not commonly used by the @foxnewspolitics account.

IV. Conclusions

Recent attacks compromising high-profile Twitter accounts had dramatic impact beyond the social network itself. Our analysis shows that these accounts exhibit highly consistent behavior over time. This consistent behavior allows us to leverage behavior-based detection techniques to protect these accounts from being compromised.

References

[1] Burger king twitter hack: Bk latest company in social media spotlight. http://articles.chicagotribune.com/2013-02-19/business/ct-burger-king-twitter-hack-0219-20130218_1_tweets-mcdonalds-hack.

[2] Fox news politics twitter account hacked, disturbing tweets appear. http://www.huffingtonpost.com/2011/07/04/fox-news-twitter-hacked_n_889590.html.

[3] U.s. stocks tank briefly in wake of associated press twitter account hack. http://allthingsd.com/20130423/u-s-stocks-tank-briefly-in-wake-of-associated-press-twitter-account-hack/.

[4] http://theonion.github.io/blog/2013/05/08/how-the-syrian-electronic-army-hacked-the-onion/, 2013.

[5] M. Egele, G. Stringhini, C. Kruegel, and G. Vigna. COMPA: Detecting Compromised Accounts on Social Networks. In *Symposium on Network and Distributed System Security (NDSS)*, 2013.

Creepic: I know who you did last summer.

Yan Shoshitaishvili
UC Santa Barbara
yans@cs.ucsb.edu

Christopher Kruegel
UC Santa Barbara
chris@cs.ucsb.edu

Giovanni Vigna
UC Santa Barbara
vigna@cs.ucsb.edu

***Abstract*—The popularity of social networks has changed the way in which we share personal thoughts, political views, and *pictures*. Pictures have a particular important role in the privacy of users, as they can convey substantial information (e.g., a person was attending an event, or has met with another person) and, because of the nature of social networks, it has become increasingly difficult to control who has access to which content. Therefore, there is a very serious potential for violations of the privacy of users when a substantial amount of pictures are accessible to one party.**

In this paper, we demonstrate a novel technique that, given a large corpus of pictures shared on a social network, can automatically determine who is dating whom, with reasonable precision. The approach combines facial recognition, spatial analysis, and machine learning techniques to determine pairs that are intimately involved. To the best of our knowledge, this is the first privacy attack of this kind performed on social networks.

We implemented our approach in a tool, called Creepic, and we evaluate it on two real-world datasets. The results show that it is possible to automatically extract non-obvious relationships between people represented in a group of pictures, even when the people involved are not directly part of a connected social clique.

I. Introduction

In the last decade, our society has experienced the explosive growth of two disruptive technologies: digital photography and the Internet. Modern users of such technology take lots of photographs of themselves and each other using their personal devices. Because these photographs are digital, and modern devices are increasingly connected to the internet, these photographs frequently end up on social networks. Facebook alone, for example, receives more than 350 million new photo uploads every day [1] while Instagram (now a Facebook subsidiary) receives 40 million.

The increasing capabilities and conveniences to share these photographs gives social networks a valuable dataset to mine them for insight into their users. For example, social networks can, and do, use modern face recognition techniques to automatically determine which of a user's friends or other users of the network are present in their photographs. This information could be used to gain insight into aspects of their users' likes beyond what the users would otherwise have provided. By accruing additional information about their users, a social network can better target advertising and improve click-through rates, in turn increasing their profits.

For example, accurately determining the relationship status of a user who is otherwise not forthcoming with such information would give a social network insight into how to properly target advertisements for such a user. If a social network was able to deduce relationship status against the wishes of such users, this would be a violation of privacy akin to un-consented web tracking. Moreover, if the information necessary to derive the relationship status of a user is available publicly (or semi-publicly to, for example, a user's Facebook friends), this privacy intrusion could be carried out by third parties.

Our intuition is that the relationship status of a user can, in fact, be determined by examining photographs in which the user in question appears. We observed that people tend to physically organize themselves differently in different situations depending on their relationships to the other people in those situations.

We created a system, *Creepic* that, given a large set of photographs, performs face detection and recognition to identify the people in each photo, extracts features of the spacial relationships between these people, and utilizes machine learning to determine which of the people are involved in an intimate relationship and to identify who their partners in such relationships are. Moreover, Creepic can oftentimes make this identification without needing a user's authorization to access their photographs. If a user appears in enough photographs (for example, if their friends grant access to *their* photographs to Creepic, and the user appears in those photographs), Creepic can still determine the user's relationship status.

We evaluate the implementation of this system against two datasets: one from a group of Facebook volunteers and another from the celebrity tracking site Zimbio.com. We show that in the case of both datasets, *Creepic* is able to identify the dating couples with a high degree of accuracy.

In short, this paper makes the following contributions:

- We develop the intuition that by analyzing photographs of a person, insight can be gleamed into their relationships with other persons.
- We develop and describe a set of statistical features that can be used by a machine learning algorithm to successfully predict user relationship status.
- We evaluate our system against two datasets with two different types of data, Facebook user data and celebrity photographs, to show its applicability in varied environments.

II. Evaluation

To determine the effectiveness of our approach, we evaluated our implementation of Creepic on two datasets: one acquired from Facebook and the other from Zimbio.com. These datasets represent the application of our approach under two different circumstances: one with the full cooperation of the users being analyzed, and one carried out without the subjects' awareness.

Facebook Dataset. Our Facebook dataset was

Dataset	Pairs	Labeled dating pairs	True positives	False positives	MCC
Facebook	2,044	130	56.3%	10.7%	0.324
Zimbio.com	7,064	1,220	57.1%	12.7%	0.416

TABLE I
THE AVERAGED RESULTS OF 10 EXECUTIONS OF 10-FOLD CROSS-VALIDATION. MCC IS MATTHEWS CORRELATION COEFFICIENT.

generated by asking volunteers to install a custom Facebook application written by us. This application first records the volunteer's relationship status and the relationship status of all of their friends. It then proceeds to download the photographs in which the user or their friends are tagged.

From the 12 users who installed our application and their 2,479 unique friends, we retrieved 175,317 photographs with a total of 87,257 unique tagged users. After matching detected faces to provided tags, we identified 142,394 user faces (of 30,531 unique users) in 85,313 photographs. 33,439 of these photographs contained more than one identified face, with a total of 90,520 identified faces present in these photographs.

We processed this dataset in 6 hours, the bulk of which was spent on face detection. Feature extraction was performed in 30 minutes, and training and classification in less than 2 minutes.

A. Celebrity Dataset

We downloaded the celebrity dataset from Zimbio.com, a website that tracks celebrity rumor and hosts paparazzi and publicity photos of celebrities. The dataset comprises 1,536,243 photographs of 2,616 celebrities and the dating history of each celebrity. After face detection and recognition, we successfully identified a total of 1,135,551 celebrity faces in 1,018,866 of the images. Out of these, 107,085 images contained more than one celebrity face, with a total of 223,770 faces celebrities being recognized in such images. The images with only a single celebrity were sampled for representative photographs. To keep memory usage of the face recognition component manageable, we randomly sampled 75 representative photographs for each celebrity.

The face detection step for this dataset took 20 hours, with a further 8 hours required for face recognition on the images containing more than one celebrity. Feature extraction was performed performed in a further 2 hours, with the training and classification requiring less than 10 minutes. Since face detection, recognition, and feature extraction is completely parallelizable between images, increasing the amount of worker nodes would linearly decrease the required number of time for all but the training and classification steps.

B. Results

We utilized Creepic to predict relationships for both the Facebook and the Zimbio.com dataset. In each case, we performed 10 executions of a 10-fold cross-validation of the dataset by randomly sampling 10% of the dating pairs for the dating portion of the training set. We aggregated these 100 classification runs and computed the Matthews correlation coefficient for the entire set. The MCC and averaged statistics are presented in Table I, followed by a brief discussion.

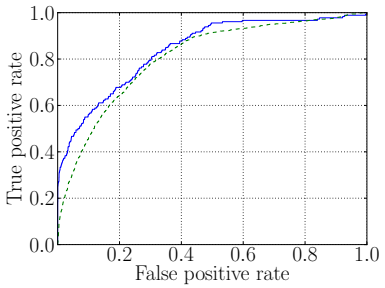


Fig. 1. The ROC curve for Creepic's relationship classification. The dotted line is for the Zimbio.com dataset and the solid line is for the Facebook dataset. The AUC for the former is 0.81 and for the latter 0.84.

Given this data, Creepic achieved a true positive rate of over 55% for both datasets, while maintaining a false positive rate of 10.7% for Facebook and 12.7% for Zimbio.com. Furthermore, a random sampling of the false positives were manually analyzed, and it was determined that a tenth of the false positives in the celebrity dataset were, in fact, actually dating pairs. For such pairs, Zimbio.com was simply missing their relationship information from the database, which caused us to mark the result as a false positive. We verified such pairs by finding other descriptions of the relationships in question on other celebrity tracking websites. For the Facebook dataset, we analyzed false positives by requesting the volunteer users to review them for accuracy. Those that responded reported about a 10% occurrence of actual, but unlabeled, relationships. This leads us to conclude that the false positive rate reported by Creepic are actually upper bounds, and the real rate could be much lower. Similarly, the actual true positive rate could be much higher.

These results are significant: Creepic is not just guessing whether a given person is dating or not. Instead, it is determining *whom* that person is dating from the pool of other users. Additionally, Creepic does this with no information other than what it extracts from images of these people.

C. Receiver Operating Characteristic

We computed the ROC curve for both the Facebook and Zimbio.com datasets. The results can be seen in Figure 1.

References

[1] I. Facebook. Facebook Inc, 2012 Form 10-k. http://bit.ly/VBYbH4.

Kwiizya: Local Cellular Network Services in Remote Areas

Mariya Zheleva, Arghyadip Paul, David L. Johnson and Elizabeth Belding
Department of Computer Science University of California, Santa Barbara
{mariya, arghyadip, davidj, ebelding}@cs.ucsb.edu

***Abstract*—Cellular networks have revolutionized the way people communicate in rural areas. At the same time, deployment of commercial grade cellular networks in areas with low population density, such as in rural sub-Saharan Africa, is prohibitively expensive relative to the return of investment. As a result, 48% of the rural population in Africa remains disconnected. To address this problem, we design a local cellular network architecture, Kwiizya, that provides basic voice and text messaging services in rural areas. In partnership with LinkNet, we deployed an instance of Kwiizya in the rural village of Macha in Zambia. In this paper we evaluate Kwiizya in-situ in Macha and show that the network maintains low delay and jitter (20ms and 3ms, respectively) for voice call traffic.**

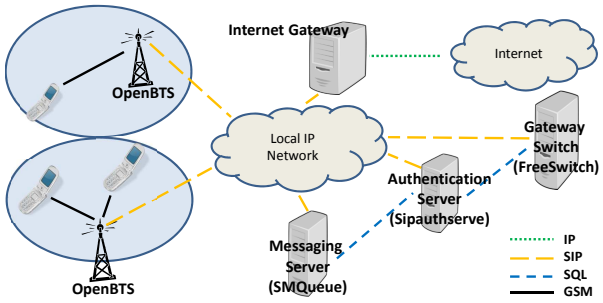


Fig. 1. Kwiizya architecture.

I. Problem Formulation

Despite low availability and high cost of cellphone services, mobile phones are critical for providing communication in remote developing regions due to limited or non-existent infrastructure. For example, access to cellphones in Macha allowed farmers to overcome what they call the "briefcase buyers problem", whereby businessmen who buy maize come to the village and often attempt to exploit farmers who are unaware of market prices, buying crops at extremely low prices. Farmers in Macha can now call the Zambian Food Reserve Agency (FRA) to get information about crops prices.

During our three week field trip to Zambia in June/July 2012, we interviewed local residents to better understand their usage and experience with voice communication technologies. All our interviewees had owned a cellphone at some point; 96% owned a phone at the time of the interview. People who owned a phone either bought the phone (76%) or obtained it as a gift. Despite the high availability of cellphones among the group we interviewed, cellphones are very expensive relative to income; 68% of phone owners who bought their phone had to save between one month and one year to afford to buy that phone. Apart from buying the cellphone, the recurring cost for air time significantly adds to the financial burden for adoption of cellphone technology.

When asked what is enjoyable about cellphone usage, 100% of our interviewees identified the opportunity to stay in touch with relatives. 80% indicated that they use the phone to obtain information, such as crop pricing for farmers, availability of goods for local businessmen and communication with regional authorities. Users with more advanced phones and data subscriptions identify the importance of staying connected to the Internet at all times. With the low rate GPRS connection in Macha, however, all users with cell phone Internet access indicated that browsing is slow and frustrating.

Our interview findings confirm the importance of cellphone technology for residents of remote areas. While the reasons for adoption of cellphone technology are not drastically different than these in the western world, the benefits for people in remote communities is much more pronounced. Unfortunately, wide availability and affordability of such technology is still severely lacking.

To address the lack of reliable and affordable cellular coverage in remote communities, we have endeavored to create a cellular access system that facilitates free local calls and SMS within a community. Our system, Kwiizya[1], leverages open-source software solutions, such as OpenBTS and FreeSwitch, to provide local voice and text messaging services. In our recent trip to Macha, we deployed two Kwiizya instantiations and gave pre-registered SIM cards to project partners. As a result, we have had the opportunity to assess usage in a fairly unregulated, in-situ scenario. In an earlier work we outlined VillageCell, an idea for providing local calls within a community [1]. In contrast, our current solution utilizes different open-source software components, offers SMS and IM-to-SMS functionality, has been fully deployed and tested in-situ in Macha, Zambia, and opened to a set of test users in Macha.

II. System Architecture

Depicted in Figure 1, Kwiizya utilizes free open-source software to provide voice and text message services. The base stations run OpenBTS, which implements the GSM

[1]Kwiizya means "to chat" in Tonga, the native language in Zambia's Southern province.

stack and communicates with the associated cellphones using the standard Um radio interface for commodity 2G and 2.5G cellphones. OpenBTS is also responsible for translating GSM messages to SIP, which allows the use of low cost generic IP backbone infrastructure as opposed to an expensive commercial-grade GSM backbone. SIP translation enables the use of free VoIP server software to serve as a Mobile Switching Center for routing calls.

Kwiizya utilizes Sipauthserve and SMQueue to handle user authentication and text messaging, respectively. SMQueue is the SIP-based equivalent of an SMSC (Short Text Messaging Central) in a commercial-grade system. As such it interfaces with OpenBTS and makes use of commodity IP networks to transmit SMS (Short Message System). SMQueue implements a store and forward SMS queue functionality that allows messages to be delivered in a delay tolerant fashion. The latter is of great importance for areas with intermittent cellphone access and electric power availability as users are often either out of range or have their cellphone powered off.

To route calls within and outside Kwiizya, we use FreeSwitch. FreeSwitch connects to OpenBTS via SIP and RTP and routes calls both in intra- and inter-BTS local scenarios. By the means of custom python scripts, FreeSwitch allows extension of the basic routing functionality to facilitate cellphone based applications. We now describe in more detail the specific extensions we have implemented.

Due to increased interest in the community in SMS-based applications, in the design of Kwiizya we are inherently interested in enabling functionality that will support such applications. Beyond simple SMS, we need functionality that supports SMS broadcast and multicast, and in particular does so through an Instant Message (IM) interface. Our IM-SMS interface enables fast typing and rapid outreach to a large set of subscribers. This interface has a wide variety of uses. For instance, a health worker could use it to notify subscribers of a change in health post hours of operation, or of the availability of particular vaccinations during a health scare. Our solution exposes an API that allows development of applications that can leverage the text broadcasting functionality to automate message generation, obviating the need of an actual person to send IM messages.

III. Evaluation

We now present results from in-situ usage of our Kwiizya deployment in Macha from a period of two weeks in July 2012. Our deployment in Macha features two RangeNetworks base stations that are interconnected through a 2.3 km directional Wi-Fi link. We evaluate critical characteristics of VoIP quality such as delay, jitter and packet loss. ITU recommendation G.114 mandates that tolerable one way delay is up to 150 ms [2]. The theoretical delay minimum that a system can provide is dependent on the used codec. Kwiizya uses GSM 6.10, for which the minimum delay is 20 ms. We evaluate these characteristics in our system by analyzing 52 VoIP sessions from Kwiizya users in Macha. Figure 2 presents our results. For each VoIP session we plot the average and

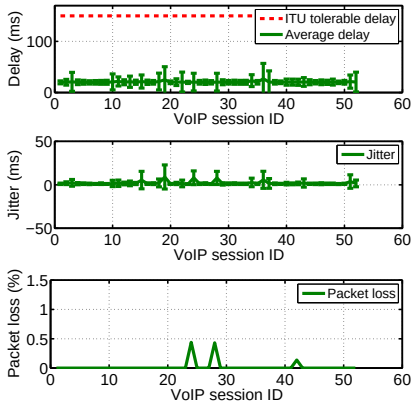


Fig. 2. Field call (a) average delay; (b) average jitter and (c) packet loss.

standard deviation of delay and jitter. A single point shows the average delay or jitter and standard deviation across all packets within that VoIP session. Both delay and jitter are well below the tolerated thresholds for VoIP. At the same time, the average delay is close to 20 ms over all VoIP sessions. Furthermore, delay and jitter do not vary much over a single VoIP session as indicated by the standard deviation bars, which shows that Kwiizya has stable performance throughout a call. Finally, we present packet loss over the 52 VoIP sessions. Only three of all sessions suffered non-zero packet loss; however, these three were all less than 0.5%, which is within the limit for satisfactory call quality.

IV. Discussion

In our work we address technical challenges in cellphone coverage for remote areas by demonstrating the feasibility of a system based on RangeNetworks hardware and open source software. Apart from the technical challenges there are economic and regulatory factors that also play an important role in alternative cellular network deployments. Economic factors concern availability of financial resources to provision power, Internet access and equipment, while regulatory factors are largely related to licensing. Our partnership with LinkNet in Macha has helped us address economic aspects of Internet access and power availability. Licensing for experimental purposes is widely unregulated in sub-Saharan Africa. Furthermore, pricing of commercial licenses is oblivious to the potential return of investment. We argue that lower pricing or subsidizing commercial licenses for economically unattractive areas such as rural areas would encourage entrepreneurs to seek alternative technologies and deploy in such areas. In this context, while Kwiizya has great technological potential to provide low-cost cellular infrastructure, the full potential of the system is yet to be realized. This realization largely depends on multi-faceted economic analysis of alternative solutions for cellphone access.

References

[1] A. Anand, V. Pejovic, E. Belding, and D. L. Johnson, "VillageCell: Cost effective cellular connectivity in rural areas," ser. ICTD '12, Atlanta, GA, 2012.

[2] "ITU-T Recommendation G.114, One-way transmission time," 2003.

Writing Groups in Computer Science Research Labs

Adam Doupé and Janet L. Kayfetz
University of California, Santa Barbara
{adoupe, kayfetz}@cs.ucsb.edu

***Abstract*—Researchers must excel at writing to effectively engage the scientific community. Clear and engaging writing advances new knowledge and increases the impact of a researcher's work.**

As developing researchers, it is essential that graduate students learn to write clearly and effectively so that their work is accessible to their peers and colleagues. In most graduate programs, students learn formal writing skills from two sources, their advisors or a writing class. We identify a third source: the graduate student peer group. In this paper, we describe how we leveraged the existing collaborative research dynamic among students in a graduate research lab and created a writing group, similar in spirit to the concept of a reading group.

I. Introduction

Ask any established researcher and she will tell you just how important writing skills are for being a successful researcher and scientist. It is through our writing that we achieve one of our main goals: To spread our ideas to our colleagues and to the general public.

Graduate students are the next wave of researchers and scientists, and it is our job as educators and advisors to instruct graduate students on all aspects of the research endeavor. Are we teaching our graduate students how to write and communicate their ideas effectively?

Traditionally, graduate students learn formal writing skills from interactions with their advisors, either in the form of red-ink editing corrections on a student's manuscript or one-on-one mentoring. A second way that graduate students learn formal writing skills is by participating in an academic writing class focused on the rhetorical demands of science writing [1].

But there is a third source where graduate students can learn formal writing skills—their peer group. Building on the culture of group analysis and discussion that already exists in the graduate student research lab, we created and introduce here the concept of a *research lab writing group,* inspired by the idea of the reading group, a common activity in many science research labs.

In the research lab writing group, graduate students not only learn the techniques of formal writing from their peers, they also learn the writing and research norms of their specific research field.

II. Goals

When developing the design of the writing group, we had multiple goals in mind.

To improve the writing of the research lab. It may seem obvious, but our goal, first and foremost, is to enhance the writing of the graduate students in the research lab.

To maximize participation. We want to maximize the number of graduate students who attend the writing group, as this will have the most impact on our lab.

To maximize interaction among the participants. We see interaction during the meeting as being different from simply showing up. It is one of our explicit objectives to encourage each lab member to share her ideas and opinions.

To develop core writing skills as well as a successful and useful approach to the analysis and discussion of texts. Our purpose is to impart the concepts and vocabulary taught in a formal academic writing class, including rhetorical positioning, audience, tone, register, coherence, readability, and so on [2], [3].

To be relevant to current writing tasks. We want our writing group to focus on current writing tasks so that we can have a direct impact on the quality of the texts produced by lab members.

To spread knowledge among the group about the ways to structure and write about research issues that are idiosyncratic to our area of Computer Science research. We want to make sure that all lab members are aware of the norms and subtleties of science discourse in our specific subfield.

III. Group Structure

In this section we describe the roles of the writing group participants along with the format of the writing group.

A. Roles

For a writing group to operate successfully and achieve all of the goals we set out in Section II, the roles of each of the participants in the writing group must be well defined.

1) Leader: The leader of the writing group is ideally a student who cares deeply about both the writing process and improving the writing of her peers. The leader should have a solid grounding in formal writing skills because it is the leader's job to teach scientific writing concepts.

2) Author: The author is a member of the writing group who is either selected or who volunteers to share her writing with the rest of the group. This role rotates for each meeting.

3) Audience: The audience is anyone who attends the writing group meeting, excluding that session's author. Nothing is required of these students before the meeting except for an open mind, a passion to improve their writing skills, and the willingness to offer helpful feedback to peer authors.

B. Format

The writing group should meet regularly, and we found that meeting weekly works best. To increase attendance, we made the time requirement manageable for the students—30 minutes.

The 30 minutes of the writing group are broken down into two different sections. The first section, lasting ten minutes, is dedicated to teaching and discussing a specific writing concept selected by the leader, such as tone, audience, abstracts, and so on.

The second phase of the writing group, lasting 20 minutes, is a group editing session of the author's writing. The author projects her writing onto a screen so that the entire audience can read the text. An important point here is that the author must prepare the text in a format that is easy to edit and change. The ability of the author to edit text quickly during the group editing session is critical so that all group members can see and evaluate in real-time the changes the author is making to the piece of writing.

Finally, at the end of a writing group session, the leader announces next week's topic, and the group chooses the author for the next week's session.

IV. Format of Group Sessions

After the initial effort to organize a writing group in our lab, our group meetings followed a clear and repeatable design.

A. Ten-minute Teaching Session

The leader begins the writing group by leading a ten minute lecture and discussion about a writing concept announced at the previous meeting.

After the leader lectures and explains the concept, there is an open discussion among the audience facilitated by the leader, focused on the concept. The conversation is meant to address any questions about the concept, to get each member of the audience thinking about how to apply the concept in their own writing, and also to identify areas where the lab's sub-field may apply the concept differently from the general scientific community.

B. Twenty-minute Group Editing Session

The group editing is supervised by the leader. The leader guides the group editing session to focus on the specific topic of the week. While it is very useful to focus on the particular "concept of the week," it is equally important that the leader also allow other topics to bubble up organically from the audience.

The leader follows a format for the group editing session. First, the leader asks the author to read her text aloud to the audience. Having authors read their own work out loud is important. The group picks up on sentences and phrasings that do not sound right.

After the author reads the text out loud to the group, the leader asks the group for comments and suggestions about how to improve the writing. However, before asking for comments, the leader must make several things clear. First, the author is in complete control of the text and any changes suggested by an audience member are up to the author's discretion. At the same time, the leader should encourage the author to experiment.

Another important thing that the leader should mention to the group is how nervous an author can feel when their writing is criticized. Therefore, the leader should ask that the group be respectful of the author's feelings when critiquing the text. At the same time, the leader should remind the author that the group is there to help, and that the author should not take comments about the writing personally.

The group discussion is so valuable because it offers the writer the unique chance to receive insightful feedback from real readers. The responsibility of the audience, then, is to give helpful suggestions that advance the writer's story, the clarity, the organization, and the readability of the text. We are particularly fond of using Weissberg's approach to keeping audience comments "as specific as possible" [4].

We advocate a thorough and detailed approach to group editing. We talk about word choice, idioms, sentence restructuring, sentence combination, adding sentences, deleting sentences, adding connections, reducing repetition, and many other topics. We also play around with different variations and keep working at finding the right phrasing and sentence structure until the author and group are happy.

C. Choosing the Next Author

An important component for the continuing success of the writing group is to have a different author present her work for the group editing session every meeting. The leader should ask, at the end of the current meeting, for volunteer authors for the next week.

V. Concluding Remarks

The goal of the scientist is to share and spread her ideas. An exceptional scientist will write her thoughts clearly and express her ideas elegantly, creating a persuasive story that is readable and interesting to her audience.

A research lab writing group, as discussed in this paper, is a novel approach to helping graduate students develop the tools necessary to refine their formal writing skills.

We hope you steal our ideas and adapt them to your own research lab. Together we can improve the writing of graduate school scientists in labs all over the world.

References

[1] J. Kayfetz and K. Almeroth, "Creating innovative writing instruction for computer science graduate students," in *Frontiers in Education Conference, 2008. FIE 2008. 38th Annual*, 2008, pp. T4F–1–T4F–6.

[2] M. desJardins, "Case Study: Teaching Research Skills to Computer Science Graduate Students," in *Proceedings of the 3rd International Conference on Education and Information Systems, Technologies and Applications (EISTA 2005), Orlando, FL*, 2005.

[3] D. G. Kay, "Computer Scientists Can Teach Writing: An Upper Division Course for Computer Science Majors," in *ACM SIGCSE Bulletin*, vol. 30, no. 1. ACM, 1998, pp. 117–120.

[4] B. Weissberg, "A Collaborative Model For Facilitating Writing Workshops," Presented at TESOL 2004, April 2004.

Submit: a mobile communication control platform

Morgan Vigil
Dept. of Computer Science University of California, Santa Barbara
Email: mvigil@cs.ucsb.edu

Waylon Brunette and Gaetano Borriello
Dept. of Computer Science University of Washington
Email: {wrb, gaetano}@cs.washington.edu

***Abstract*—In developing regions, the growing ubiquity of smart mobile technologies is unmatched by the growth of infrastructure required to support them. As developers design mobile applications for these regions, they often focus development time on handling limited communication resources. This generally manifests as a non-uniform user experience where users must actively manage communication resources through individual application interfaces. This leads to inefficient resource management and unsustainable knowledge management applications. In order to abstract communication resource management away from users and developers, we present Submit. Submit is a standalone service application for Android that interfaces with client applications through a simple API. It handles communication resources based on available networks, user imposed rules, and user history of connectivity. Submit will be deployed as part of the Open Data Kit as a free and open source tool to enable the creation of mobile data management applications.**

I. Motivation

Mobile computing devices such as tablets and smartphones embed themselves in the human experience, removing time and space constraints from knowledge production and collection [7]. As smart mobile technologies have become more accessible there has been a tremendous rise in mobile applications; mobile applications include health [4], education [1], and finance [8] applications.

While the ubiquity of mobile technology often assumes the ubiquity of infrastructure, this is not the case in developing regions. As of 2012, countries such as the Democratic Republic of the Congo and Chad had GSM penetration rates below 40% [5]. Mobile broadband subscriptions are expensive. On average, a 500 megabyte mobile broadband subscription in a developing country costs nearly 16% of the per capita gross national income.

The reality of limited infrastructure and expensive subscription plans restricts mobile applications from sharing knowledge in a fluent manner. This complication is observed on two levels. First, mobile developers cannot focus on application design without also designing a separate communication component. Second, users must handle communication preferences on a per application basis. As a result, the utility and sustainability of mobile applications suffers from a lack of unified infrastructural management.

Existing solutions lack the generality necessary for sustainable adoption in resource-poor contexts. The first approach tends to alter the underlying mobile platform, including changes to the operating system [11] or specialized networking protocols [6]. This requires specialized versions of Android that are not easily deployable en masse. The second approach is more systemic and restricts communication optimizations to a particular community [2], [10]. These solutions design local infrastructure to optimize available connectivity. While standard devices are able to take advantage of these optimizations, the optimizations are limited to the local context of the community.

In order to facilitate the creation and successful deployment of mobile applications, we present Submit. Submit is a mobile communication service that schedules network communication events based on:

- network availability,
- user imposed rules, and
- user history.

Submit's modular design allows developers to focus on designing the application, rather than the communication infrastructure supporting the application. Because it runs as a predominantly background service, Submit provides a non-intrusive way for users to impose customized rules upon the handling of communication events. Submit will be deployed as free and open source software with other tools comprising the Open Data Kit [9]. In contrast to existing solutions, Submit presents a modular approach that sits atop of the Android mobile platform and uses information from the user, the application, and the platform in order to guide communication in a manner appropriate to its context. Submit requires no modifications to the underlying Android platform and it is not restricted to a specialized infrastructure.

II. Open Data Kit

The Open Data Kit is a free and open source Android platform for mobile data collection, sharing, and management [3]. In addition to projects supported by NGOs and non-profits, ODK has also been used in several ICTD research projects, including FoneAstra, a system for managing vaccination cold chains [4]. As technicians and health workers monitor facilities and vaccination supplies along the cold chain, there are often two types of communication events required by the system: delay-tolerant updates about the state of a facility's equipment and urgent messages reporting high temperatures or disease outbreaks. We believe these scenarios correspond closely to the Sync and Message dichotomy of Submit. Part of Submit's evaluation will be based on it's performance as the communication management component of the FoneAstra system. Thus, in its initial development, Submit seeks to robustly support preexisting ODK applications.

III. System Overview

The Submit system architecture is shown in Figure 1. Submit abstracts submitted objects as one of two types:

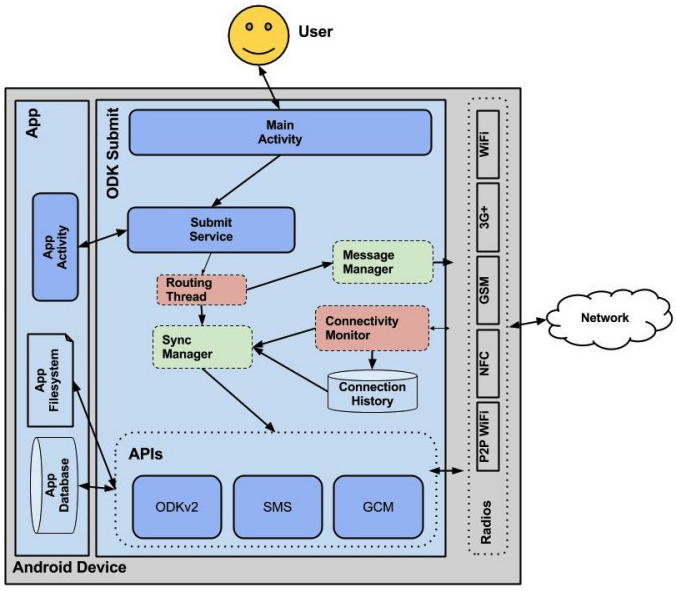


Fig. 1. Submit architecture.

1) *Messages* which contain a user formatted payload and a destination.
2) *Sync Objects* which contain a pointer to a stored data object (e.g. database object, filesystem object) and a destination.

The current Submit design handles communication over several networks, including cellular, WiFi, and NFC. Consequently, Submit implements both a client-server communication model as well as a peer-to-peer communication model.In the following sections, we describe the main interfacing components: Submit Service, the Routing Thread, Communication Managers, Connectivity Monitor, APIs, and Main Activity.

Main Activity. Users configure Submit's preferences via GUI interaction with the Main Activity.

Submit Service. Third party Activities interface with the Submit Service via the remote procedure calls `create()`, `download()`, `sync()`, and `delete()`. If an application wishes to receive feedback about the status of its submitted objects, it can register a `BroadcastReceiver` with Submit on a per transaction basis.

Routing Thread. The Submit Service starts the Routing Thread. The Routing Thread passes queued objects to the appropriate Communication Managers according to user specified metadata, such a relative data size, urgency, or priority.

Communication Managers. There are two Communication Managers, the Message Manager and the Sync Manager which handle Message and Sync objects respectively. Based on current connectivity conditions reported by the Connectivity Monitor and available API modules, the Communication Managers send submitted objects using a third party API or using the Submit Message API. If a Communication Manager is unable to send an object, the object is placed back in the Routing Thread's submission queue.

Connectivity Monitor. The Connectivity Monitor extends the BroadcastReceiver class and listens for network events, such as "WiFi enabled" or "Peer detected" events. It then passes a list of connected communication radios to the Communication Managers.

APIs. Third party APIs are supplied to Submit in JAR format. Submit is extensible and modular. Ideally, it has access to the same communication APIs that client applications leverage when they choose not to use Submit. In Figure 1, Submit uses the ODKv2 API (a RESTful synchronization API), an SMS API, and the Google Cloud Message (GCM) API. Client applications can provide their own communication APIs to Submit as long as they implement the Submit Service interface.

IV. Conclusion

Sustainable development efforts are aided by increasingly pervasive mobile data collection applications. In this work, we present Submit, a communication management platform that effectively integrates communication requests and available radio resources. This is accomplished by monitoring network availability, querying user connectivity history, and routing communication requests according to user-defined rules. The Submit communication platform is modular and easily integrated with third party Android applications using a simple API. Developers can extend Submit's communication abilities by adding their own communication JAR to the system. We believe Submit will further development efforts by making communication management a more fluid process for both developers and users.

References

[1] Dr Math – remote math tutoring using MXIT in South Africa. http://www.elearning-africa.com/eLA_Newsportal/mixing-it-with-dr-math-mobile-tutoring-on-demand/, 2013.

[2] A. Anand, V. Pejovic, E. M. Belding, and D. L. Johnson. Villagecell: cost effective cellular connectivity in rural areas. In *Proceedings of the Fifth International Conference on Information and Communication Technologies and Development*, pages 180–189. ACM, 2012.

[3] W. Brunette, M. Sundt, N. Dell, R. Chaudhri, N. Breit, and G. Borriello. Open data kit 2.0: expanding and refining information services for developing regions. In *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications*, page 10. ACM, 2013.

[4] R. Chaudhri, G. Borriello, and R. Anderson. Pervasive computing technologies to monitor vaccine cold chains in developing countries. *IEEE Pervasive Computing. Special issue on Information and Communication Technologies for Development*, 2012.

[5] GSMA. sub-Saharan Africa Mobile Observatory. http://www.gsma.com/publicpolicy/wp-content/uploads/2012/03/SSA_FullReport_v6.1_clean.pdf, 2012.

[6] K. A. Harras, M. P. Wittie, K. C. Almeroth, and E. M. Belding. Paranets: A parallel network architecture for challenged networks. In *Proceedings of the Eighth IEEE Workshop on Mobile Computing Systems and Applications*, HOTMOBILE '07, pages 73–78, Washington, DC, USA, 2007. IEEE Computer Society.

[7] K. Lyytinen and Y. Yoo. Ubiquitous computing. *Communicationsof the ACM*, 45(12):63, 2002.

[8] I. Mbiti and D. N. Weil. Mobile Banking: The Impact of M-Pesa in Kenya. Working Paper 17129, National Bureau of Economic Research, June 2011.

[9] U. of Washington. Open data kit. http://opendatakit.org/, 2013.

[10] R. K. Patra, S. Nedevschi, S. Surana, A. Sheth, L. Subramanian, and E. A. Brewer. Wildnet: Design and implementation of high performance wifi based long distance networks. In *NSDI*, volume 1, page 1, 2007.

[11] K.-K. Yap, T.-Y. Huang, M. Kobayashi, Y. Yiakoumis, N. McKeown, S. Katti, and G. Parulkar. Making use of all the networks around us: a case study in android. In *Proceedings of the 2012 ACM SIGCOMM workshop on Cellular networks: operations, challenges, and future design*, CellNet '12, pages 19–24, New York, NY, USA, 2012. ACM.

Critic: Designing a Sound Static Analyzer for Chisel

Ethan A. Kuefner, Timothy Sherwood, Ben Hardekopf
University of California, Santa Barbara
{eakuefner, sherwood, benh}@cs.ucsb.edu

John Sarracino
Harvey Mudd College
jsarracino@g.hmc.edu

I. Introduction

For hardware designers, testing is commonplace as a way to ensure that hardware designs meet their specifications. However, in high assurance settings such as flight systems, medicine, and defense, strong guarantees on the behavior of a system may be required. To achieve such guarantees using testing can be tedious for sufficiently large designs, since achieving suitable test coverage can require a potentially exponential number of test inputs. Formal static analysis provides an alternative to testing in which mathematical models are reasoned about in order to provide provable guarantees about the operation of a system. Additionally, we can design analyses to be *sound*, which means that they will only return answers which are guaranteed to be true. Thus, a static analysis-based approach to ensuring conformance of designs can remove overhead associated with exhaustive testing as well as potentially provide stronger guarantees than non-exhaustive testing.

While static analysis seems attractive, it is also difficult; in fact, Rice's theorem states that any non-trivial property of programs is undecidable. Many approaches exist to deal with this problem, but in particular, we appeal to the framework of abstract interpretation [1], which was originally designed as a technique to perform sound analyses of programs. In abstract interpretation, a formal concrete semantics for a programming language is given a mathematical correspondance to an abstract semantics that soundly approximates the behavior of programs. Abstract interpretation is well-known in the programming languages community and has even been used commercially in tools like Astrée [2], which has successfully verified C code for a variety of avionics software.

Abstract interpretation can be applied to a hardware design setting through the observation that hardware description languages are essentially a particular class of domain-specific languages (DSLs), that is, programming languages designed to target a specific application. This is the approach taken in earlier theoretical work on abstract interpretation for VHDL [3]. Our goal is to extend this insight to implement and apply a general static analyzer for Chisel. Chisel [4] is a new environment from UC Berkeley for designing and testing hardware that has been designed as an embedded DSL on top of the Scala programming language. For hardware designers, it provides advantages over existing HDLs by bringing advanced features of Scala, such as higher-order functions and object oriented programming, to the hardware design landscape. As well, it presents a good opportunity for analysis designers, since Chisel has been explicitly designed from the ground up for synthesis, unlike either VHDL or Verilog. This reins in the size of the language and allows us to more easily reason about the language and its compilation process as a whole.

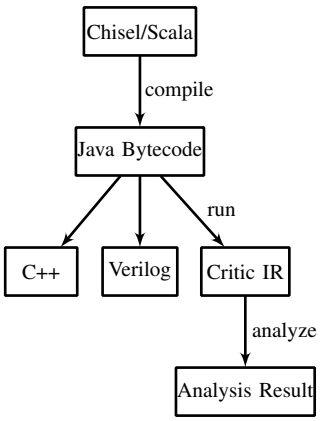


Fig. 1. Chisel/Critic workflow. The user first compiles their design and then runs it. At runtime, Chisel elaborates the design and passes it to a backend. Critic is one such backend.

This work is the latest in a series of collaborations between the PL Lab and the ArchLab towards applying techniques in programming languages to provably correct hardware design. In this note, we describe the architecture of Critic, an abstract interpreter we have been designing for Chisel, and our plans for applying it to proving particular invariants of hardware designs.

II. Design

In this section we outline first the process of extracting a hardware description amenable to analysis from Chisel, and second, the method we use to perform analysis on it.

A. Chisel

Chisel itself is simply a set of data structures which hardware designers instantiate in a particular way to build components. These data structures together represent a directed acyclic graph (DAG) of nodes such as binary operations, multiplexers, and memory reads and writes. Chisel allows extraction of C++ or Verilog from this DAG, for use in simulation or synthesis. When a designer "programs" in Chisel, she is actually programming in Scala. Figure 1 gives an overview of the Chisel/Critic workflow.

To extract a design from a Chisel program, a designer must first compile the program using the Scala compiler, producing Java bytecode. At runtime, the designer selects a "backend", either C++ or Verilog, and runs the Java bytecode, passing a flag for their selected backend. Chisel then elaborates the design, leaving a flat Chisel data structure that can be traversed for compilation. The specific backend then traverses

the data structure and outputs corresponding C++ or Verilog. Fortunately, the backend architecture is extensible, so that users can write their own backends. We take advantage of this by defining our own backend, which converts the Chisel DAG into the Critic intermediate representation (IR), which is essentially a cleaner version of the Chisel DAG which we have designed to be suitable for program analysis. The process of translating Chisel to Critic is more difficult than it may seem, since Chisel is still a research project under active development. We have had to rectify these issues through extensive reference to prose descriptions of Chisel and frequently by deferring to the actual source code.

B. Abstract interpretation

To perform abstract interpretation, we begin by formalizing the concrete semantics of the language. We use operational semantics, which specify the behavior of the language as a state transition system, potentially infinite, which operates on a collection of concrete domains representing values and components of the state. In the particular case of Critic, our state consists of a static representation of the DAG, together with a description of the current values of memories and registers, as well as the values held by the wires in the circuit and a program counter. The DAG is organized in a topological order based on data dependencies, and the program counter represents the interpreter's current node of interest in the topological order. We then formalize the state transitions in a function $\mathcal{F}\colon State \to State$ which matches deterministically on the components of the state to decide what to do. For example, if the node at the current program counter is a mux, the semantics says to read in its inputs and select one or the other based on the value of the select bit.

After we have fixed the concrete semantics, we derive an abstract semantics which finitizes the state transition system by defining a series of so-called abstract domains, together with a pair of functions called abstraction and concretization that connect the concrete semantics to the abstract semantics. The main difference between the abstract semantics and the concrete semantics is that the state transition system specified by the abstract semantics is nondeterminstic; that is, it can be viewed as a function $\mathcal{F}^\sharp\colon State^\sharp \to \mathcal{P}(State^\sharp)$. The precision of the analysis is based on the design of the abstract domains, where a key decision that has to be made regarding how much information about these sets of states, which could grow exponentially in the worst case, to discard.

III. Applications

Abstract interpretation is general enough that it can be applied to a wide variety of verification problems. In particular, we would like to use our framework to construct a general dependency analysis to annotate the circuit graph with dependency information that could be used to solve a wide variety of problems. Other areas we may target include verification of energy and thermal limits and correctness for cryptographic hardware systems.

We discuss briefly a notion of deadlock freedom for network-on-chip designs, which is provable by verifying that a dependency graph, such as one that our tool could produce, is cycle-free. Consider a simple packet-switched network with a ring topology on four nodes, where each node X has a register into which packets are injected by the controller X_i, and a register for use in forwarding packets X_o. The network can make one move per clock cycle; that is, exactly one node can forward a packet on any given cycle, and "starting" a packet (that is, extracting it from the point where it was injected) is the highest-priority move. We will refer to these nodes as A, B, C, and D, and assume that packets are injected to a particular node by some controller, with a particular destination specified. Table I specifies four packets to be forwarded. In the first four cycles, packets 1–4 move into B–A's forwarding queues respectively. At this point, no more moves can be made, since all the forwarding queues are full, and this is what we define as deadlock. A number of fixes exist for addressing this, including for example wormhole routing, in which each node receives an additional queue and the routing policy is modified to use this additional queue to avoid conflicts. It is of interest whether such solutions are correct, and it is a goal of our analyses to determine this. In other words, given a strategy for *deadlock avoidance* and an implementation of it in a hardware design, we would like to be able to prove that the particular implementation renders the design *deadlock free*. This will require formulating a particular set of invariants that will hold over all runtime configurations for the design in question.

#	Src	Dst
1	A	C
2	B	D
3	C	A
4	D	B

TABLE I. Packet list for routing example

IV. Conclusion

In this extended abstract we have detailed Critic, a design for an abstract interpreter based on Chisel, a new hardware construction language from UC Berkeley. We described our process for working with Chisel, as well as aspects of the design of our analyzer. We have detailed a specific client for our analyzer, deadlock freedom, as well as other clients with which to validate the analyzer.

Acknowledgements

We would like to thank Hassan Wassel for helpful discussions about hardware properties and their verification.

References

[1] P. Cousot and R. Cousot, "Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints," in *Proceedings of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, ser. POPL '77, 1977, pp. 238–252.

[2] http://www.astree.ens.fr/.

[3] C. Hymans, "Checking safety properties of behavioral VHDL descriptions by abstract interpretation," in *In 9th International Static Analysis Symposium (SAS'02) (2002*. Springer, pp. 444–460.

[4] J. Bachrach, H. Vo, B. Richards, Y. Lee, A. Waterman, R. Avizienis, J. Wawrzynek, and K. Asanovic, "Chisel: Constructing hardware in a Scala embedded language," in *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE*, 2012.

On the Self-similarity of Social Network Dynamics

Qingyun Liu, Xiaohan Zhao, Stephanie Smith,
Ben Y. Zhao and Haitao Zheng
Computer Science, UC Santa Barbara
{qingyun_liu, xiaohanzhao, ravenben,
htzheng}@cs.ucsb.edu, snsith90@gmail.com

Walter Willinger

AT&T Labs Research
walter@attt.research.com

I. Introduction

Analysis and modeling social network dynamics is key to making accurate predictions, which is in turn critical to applications such as network infrastructure design, resource allocation, and anomaly detection. Unfortunately, we still lack a detailed understanding of dynamics in online social networks (OSNs), which is a direct reflection of the limitations of currently available datasets. While accurate models of network dynamics require time-stamped traces of network events, available datasets today are mostly gathered as one or more *static* snapshots of networks undergoing change, leading to unsatisfactory models that produce only *logical* sequences of events without notions of absolute clock times.

We have recently obtained access to a large, detailed trace of network events in Renren social network [1]. In the first step of developing our time-based model that predict growth in OSNs, *e.g.* new users and social links (also called edges), we are reminded of lessons from network traffic modeling that show network traffic events to exhibit "self-similar" properties, which cannot be accurately captured using traditional models.

In this paper, we describe our efforts to search for and identify potential self-similar properties in the edge creation event trace of social network evolution. We study a sequence of about 186 million events capturing the creation of all edges over a period of 12 months from Jan. 2007 to Dec. 2007, using a range of techniques including the R/S analysis, variance fitting and a wavelet-based method. Our work produces three key findings. *First*, edge creation process of our dataset displays a typical diurnal pattern even after removing the impact of node arrivals. *Second*, local fluctuations of the diurnal component exhibit properties consistent with self-similarity at small time scales. *Finally*, the wavelet-based method is highly robust to detecting self-similarity in the presence of underlying deterministic trends, which argues for using it as the preferred method of self-similarity detection.

To the best of our knowledge, our work is the first to study the presence of self-similarity in the dynamics of complex networks. Our findings mean that we can build accurate event models of OSN dynamics using a combination of long-term deterministic trends and a short-term self-similar component explaining the random fluctuations around it. This in turn has potentially significant implications on a range of applications such as resource allocation in OSNs.

II. Background

In this section, we introduce briefly the notion of self-similarity, and describe the Renren dataset we used.

Self-similarity. For a time process, self-similarity refers to the scale invariance behavior [2], *i.e.* statistical properties of this stationary process look similar at different time scales. An effective and commonly used metric to measure the existence of self-similarity is the Hurst parameter H. When H falls in the range of (0.5, 1), the process exhibits self-similarity. Ideally, the statistical properties of a self-similar process should stay invariant across all time scales. In reality, such properties often exist at small time scales, but break down at large time scales due to periodic patterns or finite lifetimes. Thus, for modeling it is not only important to identify self-similarity, but also the range of time scales for which it is visible in the dataset.

Dataset. Our analysis uses an anonymized dataset from Renren, the largest and oldest Chinese OSN with more than 220 million users. The dataset contains a time-stamped (down to the second) trace of the creation of all nodes (19,413,375) and all edges (199,563,976) over a 25-month period from Nov. 21, 2005 (the launch of Renren) to Dec. 31, 2007. In Dec. 2006, an unusual event happened – Renren merged with another OSN, leading to a burst of edge growth around that time. To our best knowledge, this is the largest time-stamped data on social network evolution.

III. Experiment

Our goal is to investigate if Renren's edge creation process displays properties consistent with self-similarity, and if so, at what time scales.

A. Initial Analysis

A key challenge we face is identifying and isolating the impact of deterministic trends in the edge creation data. As a first step, we limit the impact of new node arrivals on edge creation, by focusing our analysis on edge created between members of a fixed user population. Specifically, we prune our dataset to include only existing users as of Dec. 1, 2007, and study all edge creation events between them in Dec. 2007. We choose this month period because it is late enough in the history of Renren that it avoids the initial exponential network growth and the abnormal burst of edge growth during the merge event in Dec. 2006. We remove this restriction and extend our analysis to all edge creation events in subsection *C*.

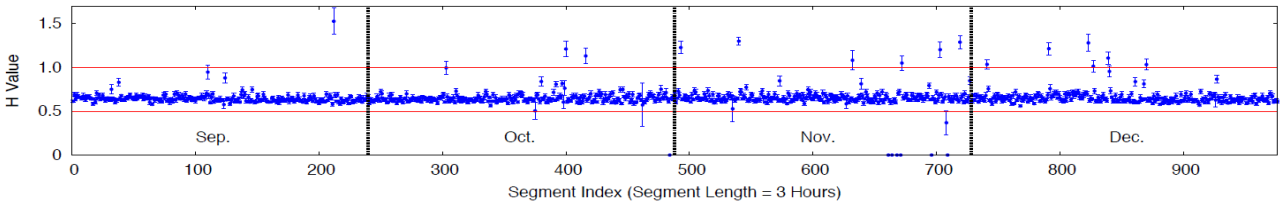


Figure 1: The estimate of H of all the disjoint 3-hour segments between Sep. – Dec. 2007, performing wavelet analysis on the entire dataset without sampling. The result aligns with those with sampling.

Our initial analysis applies two most popular methods used to estimate the Hurst parameter H, *i.e.* variance analysis and R/S analysis, in addition to directly visualizing the raw data.

By plotting the number of new edges created in each second over the one-month period, we clearly observe a diurnal pattern in the edge creation process. This non-stationary behavior precludes direct analysis on self-similarity. However, both variance and R/S analysis show that our data seem to lose consistent with self-similarity only when the time period exceeds 10^4 seconds (3 hours). To confirm the possibility of a self-similar structure in the local fluctuations of the diurnal component, we perform variance and R/S analysis on each of the disjointed 3-hour long segments over the entire month and compute the corresponding H value. For all the 248 segments, more than 98% have estimated H within (0.5,1), with mean 0.8867 and variance 0.0108 for variance analysis while mean 0.6752 and variance 0.0006 for R/S analysis. These results strongly suggest that local fluctuations at time scales of 3 hours exhibit properties consistent with self-similarity.

B. *Wavelet-based Analysis*

Potential issues of reliability for both variance and R/S analysis are that they sometimes produce poor fitting to the data and therefore a questionable estimate of H. These errors can be attributed to their easily affected "eyeballing" approach in highly variable data. In contrast, a wavelet-based method [3] offers a more rigorous analysis on the data's scaling property by removing the impact of deterministic trends as well as providing a confidence interval for each H estimate. Here we apply this wavelet-based method to confirm preliminary results of potential behaviors of self-similarity on our sampled dataset.

We divide the sampled dataset into segments of lengths between 3 and 12 hours, and apply wavelet analysis to each segment. In our analysis, we refer to a segment as "abnormal" if its H estimate (including the 95% confidence interval) does not completely fall within the self-similar range (0.5,1). Our analysis leads to three key findings:

Self-similarity in Short Time Scales. Our analysis confirm that at small time scales (around 3 hours), the large majority (98%) of data segments consistently exhibit self-similarity scaling behavior with H centering around 0.63. We also looked at different segment compositions by shifting their start time, and observed similar results.

Weaker Self-similarity at Larger Time Scales. We observe that the ratio of abnormal segments increases quickly from 2.02% for 3-hour segments, to 8.06% for 6-hour segments, and 32.36% for 12-hour segments. This confirms our earlier conclusion that at large time scales, the edge creation process displays properties less consistent with self-similarity, possibly due to the increasing impact of periodic processes.

Pattern of Abnormal Segments. We find that abnormal segments are randomly distributed across days. Within a day, their appearance correlates strongly with the frequency of user activities, e.g. most likely to occur during 6pm-9pm but rarely at 3am-6am. We also find that abnormal segments are caused by sudden changes of edge creation which might be caused by Renren's new features, localized failures or ad promotions. We are working with Renren to further confirm these hypotheses.

C. *Analysis Without Sampling*

Finally, we expand our analysis to the entire unsampled dataset to explore whether the observed behavior consistent with self-similarity on sampled data is still present after including new nodes with rapid edge growth.

We first consider unsampled dataset in the month of Dec. 2007. Results show 97% of the 3-hour segments display properties consistent with self-similarity with mean H=0.65, which aligns with our prior analysis on the sampled dataset.

Next we examine all the edge creation events in the year of 2007. Again we observe the same conclusion consistently across the entire year: 97% of the 3-hour segments display properties consistent with self-similarity, with the mean of H at 0.64. Figure 1 shows the H result for Sep.-Dec. 2007 (due to the space limit), which is stable across all the months.

IV. DISCUSSION

From a modeling perspective, our observation of self-similar scaling behavior in Renren suggests a new two-stage approach to model edge creation process in social network dynamics. In stage one, "primary" events (i.e. individual user arrivals) are modeled with a non-homogeneous process that accounts for nonstationary features such as exponential growth and diurnal patterns. In stage two, "secondary" events (i.e. edge creation for individual users) are modeled. For example, the number of edge created per user is drawn from a certain heavy-tailed distribution and individual users generate edges in some fashion over time. Mathematically, the aggregation of such secondary events across a large number of users will exhibit self-similar scaling [4].

REFERENCES

[1] ZHAO, X., SALA, A., WILSON, C.,WANG, X., GAITO, S.,ZHENG, H., AND ZHAO, B. Multi-scale dynamics in amassive online social network. In Proc. of IMC (2012).

[2] LELAND, W., TAQQU, M.,WILLINGER, W., AND WILSON, D. On the self-similar nature of ethernet traffic. IEEE/ACM Transactions on Networking (1994).

[3] ABRY, P., AND VEITCH, D.Wavelet analysis of long-range-dependent traffic. IEEE Transactions on Information Theory (1998).

[4] RESNICK, S. I. Heavy-tail phenomena: probabilistic and statistical modeling. Springer-Verlag, 2007.

Say My Name, Say My Name: User Mentioning on Facebook

Saiph Savage
4 Eyes Lab, UCSB
Email:saiph@cs.ucsb.edu

Andres Monroy-Hernandez
Microsoft Research
Email: andresmh@andresmh.com

Leif Singer
University of Victoria
Email: lsinger@uvic.ca

Tobias Hollerer
4 Eyes Lab, UCSB
Email:holl@cs.ucsb.edu

***Abstract*—Online social networks have become a place to broadcast personal exchanges, political opinions, and cultural artifacts. Many users tag or mention others—sometimes as a tribute, sometimes to get visibility. However, the social implications of these *broadcasts with user tagging* have so far been understudied. In our qualitative study, we uncover how users experience broadcasts that tag them and other Facebook users. By combining surveys and semi-structured interviews, we find that users enjoy being mentioned in content, as it signals they have connected well with friends. Users who tag others strive for visibility and higher quality social interactions with their network. However, some tagged users feel a loss of control and suspect that others define their interests and social interactions. Our results provide design implications for supporting broadcasts with user tagging in social media tools and generally serve to better understand this phenomenon.**

I. Introduction

Tagging or mentioning other users in content broadcasted to large audiences began on Twitter in late 2006. Users began adopting the *at sign* ("@") as a way to direct public posts to particular users. Its popularity lead Twitter to add native support in 2008—since then, any instances of an @ followed by a username link to the respective user's profile. In 2009, Facebook granted users the ability to tag friends in their posts. Broadcasts with user tagging are now a common practice in online social networks (OSN).[1]

Since the emergence of user tagging in social media, several studies have investigated different aspects of the phenomenon. Users tag others for conversations and for acknowledging others. Sometimes users will value being mentioned, but other times they may feel exploited—as evidenced by the recent lawsuit over users' names appearing in online advertisements. However, so far to our knowledge no study has explored users' experiences with tagging and being tagged in depth.

The majority of research about user tagging has focused on creating interactive interfaces through which users can tag and augment the information a system has about others, primarily to improve the system's recommendations [1]. But, there is a lack of understanding how tagging and linking data to others in broadcasts affects those involved.

Tagging on social media has gotten to the point where it can even be compared to grafitti. People in graffiti want to be seen in their city, they are therefore creative and come up with ways to tag their city with their content, getting away from the police, getting up on dangerous buildings, likewise Facebook taggers come up with ways to distribute their content, and make it be seen by many.

Facebook and Twitter represent graffiti in that both represent normal, average people making their voice heard through creative efforts that are not professionally supported but very much public. A Facebook or Twitter presence can lead to money but it does not itself pay, nevertheless everything you say is public to all your friends and colleagues. Grafitti artists are not paid for their work (though some are through mural works they have been contracted for - like social media, grafitti can lead to work but is not work in and of itself) but nevertheless have a significant public impact by what they say and illustrate publicly.

Since the data linked to a person can affect the impressions they make on others [2], we need to better understand users' experiences related to tagging in broadcast messages. As the practice becomes supported on more and more platforms, it becomes important to understand the diversity of experiences emerging from this interaction. This will help us comprehend the larger issues regarding user identity, privacy, and the type of public social interactions users want. In our qualitative study, we use surveys and semi-structured interviews to explore user experiences with tagging and being tagged in broadcast messages on Facebook.

We find that users tag others in broadcasts to profit from their reputation, identity, and social graph. Tagged users enjoy the activity, as it signals they are building stronger relationships. However, they also struggle with feeling that their friends are defining their interests and social interactions. Users following their friends' tagged data expressed that even though it occasionally bordered on spam, they enjoyed viewing the content as it provided serendipitous discoveries.

II. Background and Related Work

This section provides more background to our contribution and puts it into context with previous research on the different sharing and communications modes that users of social network sites employ.

A. A Spectrum of Sharing Modes

OSN users commonly have contacts from different life facets, e.g. friends from school or college, work colleagues, and others. However, users' different life facets can easily get mixed when publicly broadcasting content: a *context collapse*

[1] work in progress, please do not cite and redistribute.

occurs. Friends and family may receive work-related content that is irrelevant to them. Similarly, broadcasting personal content to work contacts may at times be inappropriate.

Users engage in a spectrum of sharing modes to overcome context collapse. This spectrum can range from (a) *targeted sharing*: sharing specific data with particular individuals—examples are email and private messages [3]; to (b) *broadcasts*: sharing content generalizable enough to be appropriate for all audiences.

Within this spectrum, the work of Kairam et al. [3] studies *selective sharing*, the sharing of content with certain lists or circles such as work, family, or those pertaining to a specific hobby. The authors find that users engage in selective sharing typically for evangelism purposes, to enter conversations, or to ask questions. The work also introduces a study on targeted sharing, comparing its usage with selective sharing and public broadcasting. Bernstein et. al. studied targeted sharing in email, and found that the smaller the audience, the more the recipient felt the message was personalized.

In our study, we explore a sharing mode that is a hybrid of targeted sharing and broadcasting: *broadcasting with user tags*. When a user tags others in broadcasted data, the content is shared directly with the person tagged—similar to targeted sharing—and with additional audiences, similar to a broadcast. This hybrid sharing mode has yet to be studied in detail, especially given its interesting and complex cultural dynamics. We plan on using the findings of this study to build better and more robust sharing systems [4].

III. Study Design

We conducted a survey and interview study with active Facebook users to understand users' experiences with broadcasting and tagging. Participating users had either: (a) tagged other Facebook users in a post shared with their network ("taggers"); (b) been tagged by others in posts shared with their network ("tagees"); (c) witnessed their friends being tagged in posts shared with their network ("viewers").

We conducted a survey to identify different types of Facebook posts where people are tagged. The outcome of this initial inquiry provided a starting point to ask a larger pool of users about their experiences with broadcasts and tagging, the frequency of being involved in such interactions, and their enjoyment levels for these interactions. We then categorized their responses and quantified their reported frequencies and enjoyment levels. Finally, we used qualitative coding to create a taxonomy of experiences about broadcasts with user tagging from our data.

IV. Findings

Content curators want to understand their audiences and involve them with their content. Tools specialized on finding relevant users to tag could support these needs. Such a tool could present different multi-faceted views that let the user study and compare the social features of these potentially relevant users.

Our results also indicate that viewers would benefit from systems that automatically detect and highlight content involving their friends, generated by interesting strangers. Such an approach would help provide serendipitous discoveries. It could also help users understand the social dynamics of other groups, and identify the friends that could act as bridges. Such a tool could also be leveraged by the spatial constraints of the contacts, to provide better situation awareness [5].

V. Conclusions and Outlook

We used Facebook as a medium to investigate how people experience broadcasts that mention other users.

References

[1] Michael Bernstein, Desney Tan, Greg Smith, Mary Czerwinski, and Eric Horvitz. 2009. Collabio: a game for annotating people within social networks. In Proceedings of the 22nd annual ACM symposium on User interface software and technology (UIST '09). ACM, New York, NY, USA, 97-100

[2] Barash, V.; Ducheneaut, N.; Isaacs, E.; Bellotti, V. (2010). Faceplant: impression (mis)management in Facebook status updates. Proceedings of 4th International AAAI Conference on Weblogs and Social Media (ICWSM); 2010 May 23-26; Washington, DC.

[3] Sanjay Kairam, Mike Brzozowski, David Huffaker, and Ed Chi. 2012. Talking in circles: selective sharing in google+. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12). ACM, New York, NY, USA, 1065-1074. DOI=10.1145/2207676.2208552 http://doi.acm.org/10.1145/2207676.2208552

[4] A.G. Forbes, S. Savage, and T. Hollerer. Visualizing and Verifying Directed Social Queries. IEEE Workshop on Interactive Visual Text Analytics. Seattle, WA, USA, 2012.

[5] Im feeling loco: A location based context aware recommendation system, NS Savage, M Baranski, NE Chavez, T Hllerer - Advances in Location-Based Services, 2012

http://gswc.cs.ucsb.edu