GSWC 2010

Proceedings of

The Fifth Annual Graduate Student Workshop on Computing

October 8th, 2010
Santa Barbara, California

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

Organized By

Bryce Boe, Chair
Lara Deek, Vice-Chair
Wei Tang, ECE Co-Coordinator
Jennifer Chen, ECE Co-Coordinator
Mike Wittie, Proceedings Coordinator
Nichole Stockman, General Committee
Petko Bogdanov, General Committee
Ceren Budak, General Committee
Sudipto Das, General Committee
Aaron Elmore, General Committee
Luca Foschini, General Committee
Bita Mazloom, General Committee
Gianluca Stringhini, General Committee
Jonathan Ventura, General Committee
Christo Wilson, General Committee
Fred Chong, Faculty Adviser

Thanks to

Platinum Supporters

Gold Supporters

Keynote Speakers

Tajana Šimunić Rosing, Assistant Professor, UCSD

Tajana Šimunić Rosing is currently an Assistant Professor in Computer Science Department at UCSD. Her research interests are energy efficient computing, embedded and wireless systems. Tajana's work on event driven dynamic power management laid the mathematical foundations for the engineering problem, devised a globally optimal solution and more importantly defined the framework for future researchers to approach these kinds of problems in embedded system design. Her recent results demonstrate the importance of joint power and thermal management in multicore server systems in order to minimize the overall energy cost. Furthermore, she developed a novel class of proactive thermal management policies that can lower the incidence of hot spots in multicore processors by up to 60% with no performance impact. Her current work is focused on developing energy efficient scheduling policies for virtualized server environments and on energy efficiency in population area healthcare networks. From 1998 until 2005 she was a full time research scientist at HP Labs while also leading research efforts at Stanford University. She finished her PhD in EE in 2001 at Stanford, concurrently with finishing her Masters in Engineering Management. Her PhD topic was dynamic management of power consumption. Prior to pursuing the PhD, she worked as a senior design engineer at Altera Corporation. She obtained the MS in EE from University of Arizona. Her MS thesis topic was high-speed interconnect and driver-receiver circuit design. She has served at a number of Technical Paper Committees, and is currently an Associate Editor of IEEE Transactions on Mobile Computing. In the past she has been an Associate Editor of IEEE Transactions on Circuits and Systems.

Úlfar Erlingsson, Security Research Manager, Google

Úlfar Erlingsson leads security research efforts within Google Research. Previously, he has been a researcher at Microsoft Research, an Associate Professor at Reykjavik University, Iceland, and led security technology at two startups: GreenBorder and deCODE Genetics. He holds a PhD in CS from Cornell.

Discussion Panel

Lingli Zhang, Software Engineer, Microsoft

Dr. Lingli Zhang is currently a Software Engineer at Microsoft, Technical Computing Group. Her research interests include programming language and compiler, virtual execution environment and programming support for parallel computing. Her last significant endeavor was contributing to the Microsoft STM.net release, a software transactional memory implementation on .NET. She currently works on a new research incubation project on parallel programming support in C++. Dr. Zhang received her M.S. from Zhejiang University, China, and her Ph.D. in computer science from University of California, Santa Barbara. Her dissertation is about "exploiting adaptation in a Java Virtual Machine to enable both programmer productivity and performance for heterogeneous devices". She has published a series of papers on code memory management and supporting Futures in Java Virtual Machine runtime.

Joe Alfaro, Sr. Director of Engineering, Citrix Online

Joe Alfaro is Sr. Director of Engineering at Citrix Online. He is responsible for the engineering teams that produce all COL products including GotoMyPC, GotoAssist, GotoMeeting, GotoWebinar, GotoTraining and GotoManage. Prior to this Joe was the VP of Engineering for Velosel Corporation, a SaaS supply chain management company. Prior to that Joe was VP of Product development for Interbase Software Corporation which produced relational database software. Before that, Joe held a variety of technical and management positions at Symantec, Borland, Apple and a number of small startups.

Jon Walker, CTO, AppFolio

Jon is a serial entrepreneur and is working on his third successful startup, AppFolio, with former UCSB professor and founder of Citrix Online, Klaus Schauser. AppFolio is the new generation of Software as a Service company. Prior to AppFolio, Jon has been the CTO of two successful startups. The first was Miramar Systems. During his tenure, he led the development and quality assurance organization team and was responsible for the creation and delivery of products. The software developed under his direction has been deployed to over 20 million computers worldwide. Miramar was sold to Computer Associates in 2004. The second was Versora, which provides open source Systems Management software. Versora was sold to Kaseya in 2006. Jon has also been a senior technologist for Nortel Networks and Xing Technology Inc. (sold to Real Networks), a Contributing Editor to LinuxWorld Magazine and he is the inventor of multiple patents. Jon also teaches Software Engineering as an adjunct professor at Westmont College in the Computer Science department.

Table of Contents

Posters

Eliminating Timing and Termination Leaks

Vineeth Kashyap, Ben Hardekopf
Department of Computer Science
University of California, Santa Barbara
{benh, vineeth}@cs.ucsb.edu

Ben Wiedermann
Department of Computer Sciences
University of Texas at Austin
ben@cs.utexas.edu

***Abstract*—Secure information flow is an important property for guaranteeing the privacy and integrity of information. Most programming languages that enforce secure information flow prevent *explicit* and *implicit* leaks that arise from data and control dependencies. However, leaks that arise from covert channels such as timing and termination can expose an arbitrary amount of information and are difficult to prevent at the programming-language level. Prior efforts to address such covert channels do so by severely restricting the expressiveness of the language, such as forbidding branches or loops whose conditions rely on secret information.**

In this paper, we outline programming language mechanisms that, given a program without any implicit or explicit leaks, transform the program into a version that also contains no timing or termination leaks. The technique is based on slicing the program into a number of sub-programs, one per security level, and scheduling the sub-programs in a manner guaranteed to eliminate illegal flows. Our technique is able to make stronger security guarantees than any existing language-based technique, and unlike existing work it does so without imposing additional restrictions on the language.

I. Introduction

Secure information flow guarantees the privacy and integrity of data, thereby prohibiting an attacker from learning secret information (privacy) or injecting untrusted information (integrity). For example, a financial application might have access to a user's private financial data as well as the ability to send messages over the network to access public financial data such as stock quotes. This level of access is necessary for the application to function—it cannot be denied either resource. However, the user would like some guarantee that a malicious application cannot use its network access to make the user's private information public.

A common guarantee that secure information flow can provide is called *noninterference*. This guarantee states that the behavior of publicly observable events (e.g., messages sent over a network) cannot be influenced by private data (e.g., a user's private information). The task of secure information flow is to identify *information channels*—mechanisms that are able to transmit information—and prohibit *leaks* (information flows along those channels that violate noninterference).

The most prevalent information channels addressed by existing work are *explicit* and *implicit* channels, corresponding to data and control dependencies in a program. However, the most difficult types of information channels to control are *covert channels*; these channels are not intended by their nature to transmit information, but they can be subverted for this task. This paper is concerned with two types of covert channels called *timing* and *termination* channels. These channels attempt to leak information by modifying the timing or termination behavior of a program based on secret information; an attacker who is able to observe this behavior can then deduce something about the values of the secret data. Timing and termination channels have been used in practical attacks against privacy (e.g., in web applications [2]) and are able to leak an arbitrary amount of information.

In this paper we improve on the state of the art by contributing a novel program transformation capable of completely eliminating timing and termination leaks from a program that has no explicit and implicit flows. Our key insights are that: (1) if a program exhibits no illegal explicit or implicit flows then the computations at the various security levels are essentially independent of one another; (2) such a program can safely be decomposed into a set of independent sub-programs; and (3) these sub-programs can then be scheduled to execute in a way that prohibits timing and termination leaks.

Whereas previous techniques remove timing and termination leaks by imposing severe restrictions on the programming language—for example by forbidding branches or loops whose conditions rely on secret information—our technique makes stronger security guarantees than any existing language-based technique, without imposing additional restrictions on the language.

II. Security Models and Noninterference

The information flow policy of a program is defined using a lattice $(\mathcal{L}, \sqsubseteq)$ where $\mathcal{L}$ is a set of security classes and $\sqsubseteq$ is a partial order indicating relative secrecy among those classes. The terms "high information" and "low information" indicate relative position in the lattice, high being "more secret".

Our definition of secure information flow uses the notion of *noninterference*, which states that values of variables at a given security level $\ell \in \mathcal{L}$ can only influence the values of variables at any security level that is lower than or equal to ℓ in the security lattice. Based on the definition of *influence*, various security models can be derived.

The simplest definition of *influence* considers only values of variables: a program is noninterfering if changing the values of variables at level $\ell' \not\sqsubseteq \ell$ cannot affect the values of variables at level ℓ either directly via data dependencies (called *explicit channels*) or indirectly via control dependencies (called *implicit channels*). More sophisticated definitions of *influence*

also account for *timing channels* (the ability of values at level ℓ' to affect *when* the values at level ℓ are computed) and *termination channels* (the ability of values at level ℓ' to affect *whether* the values at level ℓ are computed, via nontermination or abnormal termination). Both timing and termination aspects of the security model have a dimension of *sensitivity*, which describe the assumptions that the model makes about the attacker's strength.

The three security models with regard to timing are: (a) timing-insensitive model (TIME-I) which assumes that an attacker has no means to time program execution (b) weakly timing-sensitive model (TIME-WS) which assumes time is incremented by a fixed, constant amount at each program step (c) strongly timing-sensitive model (TIME-SS) which assumes that an attacker is able to time program execution using "wall-clock" time and thus accounts for architectural features such as caches and branch predictors.

The three security models with regard to termination are: (a) termination-insensitive security model (TERM-I) which makes security guarantees under the assumption that a program terminates normally (b) weakly termination-sensitive model (TERM-WS) which assumes attacker can observe termination but ignores abnormal termination (c) strongly termination-sensitive model (TERM-SS) which acknowledges the possibilities of both non-termination and abnormal termination.

Using these descriptions we can distinguish three types of noninterference properties: (a) Insensitive noninterference: $\langle$TIME-I, TERM-I$\rangle$ (b) Weakly-sensitive noninterference: $\langle$TIME-WS, TERM-WS$\rangle$ (c) Strongly-sensitive noninterference: $\langle$TIME-SS, TERM-SS$\rangle$.

In several related previous work [1], type systems have been proposed that guarantee insensitive noninterference. In the next two sections, we outline a mechanism that takes an insensitively noninterfering sequential program and transforms it to remove timing and termination leaks. The degree of sensitivity exhibited by the transformed program is a function of that program's security lattice.

III. Sketch of Security Transformation

The concept behind the security transformation is to statically slice an insensitively noninterfering program with n security levels into n separate sub-programs, each of which computes the values at its own security level. The slicing transformation is applied to the program n times, once per security level in a given lattice $\mathcal{L}$. A slice at level ℓ computes only the values at its own security level. The slicing transformation treats ℓ as a low security label and elides any computation or value that has a higher or non-comparable label. For simplicity, we refer to both high and non-comparable computations and values as simply "high".

Thus the transformation causes the low computation to skip past any high computation and go directly to the next low computation, if any. In the original program, it is possible that the high computation sometimes diverges and never returns to the low computation. But the transformation, coupled with the scheduler discipline, guarantees that the low computation can never observe the high computation's timing or termination behavior.

IV. Sketch of Secure Scheduling

The goal of secure scheduling is to execute n insensitively noninterfering sub-programs in a way that eliminates timing and termination leaks. Our scheduler: (a) handles an arbitrary lattice, (b) enforces strongly-sensitive noninterference among comparable security levels and weakly-sensitive noninterference among non-comparable security levels.

The scheduler prevents leaks among comparable sub-programs by executing them in increasing order, according to the security lattice, and hence no sub-program can observe timing or termination channels from higher sub-programs. Thus the scheduler guarantees strong noninterference between comparable security levels.

A sub-program is ready to execute when all its comparable, lower sub-programs have terminated. If multiple non-comparable sub-programs are ready to execute, the scheduler multiplexes them in a fair manner. Since the non-comparable sub-programs being multiplexed together are sharing hardware resources, timing leaks can occur through caches, branch predictors etc. Therefore, the scheduler can only guarantee weak noninterference among non-comparable security levels, unless it receives support from architectural level. During multiplexing, the scheduler is essentially interleaving the computations of multiple sub-programs, and hence care must be taken to provide the guarantee of weak non-interference among non-comparable security levels. Naive schedulers that vary the number of sub-programs being multiplexed can leak termination information, so do schedulers that force one sub-program to wait for another non-comparable sub-program to terminate as a part of their scheduling strategy.

References

[1] Andrei Sabelfeld and Andrew C. Myers. Language-based information-flow security. *IEEE Journal on Selected Areas in Communications*, 21(1):519, January 2003.

[2] E. W. Felten and M. A. Schneider. Timing attacks on web privacy. *ACM Conference on Computer and Communications Security*, pages 25-32, 2000.

Efficient and Scene-Adaptive Capture of Focal Stacks

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

Natasha Gelfand, Marius Tico, Kari Pulli
Nokia Research Center, Palo Alto
{natasha.gelfand, marius.tico, kari.pulli}@nokia.com

***Abstract*—All-in-focus imaging is a computational photography technique that produces images free of defocus blur by capturing a stack of images focused at different distances and merging them into a single sharp result. Current approaches assume that images have been captured offline, and that a reasonably powerful computer is available to process them. In contrast, we focus on the problem of how to capture such input stacks in an efficient and scene-adaptive fashion. Inspired by passive autofocus techniques, which select a single best plane of focus in the scene, we propose a method to automatically select a minimal set of images, focused at different depths, such that all objects in a given scene are in focus in at least one image. We aim to minimize both the amount of time spent analyzing the scene and capturing the images, and the total amount of high-resolution data that is captured. The algorithm first analyzes a set of low-resolution sharpness measurements of the scene while continuously varying the focus distance of the lens. From these measurements, we estimate the final lens positions required to capture all objects in the scene in acceptable focus. We demonstrate the use of our technique in a mobile computational photography scenario, where it is essential to minimize image capture time (as the camera is typically handheld) and processing time (as the computation and energy resources are limited).**

I. Introduction

The size of a camera's aperture provides a trade-off between the depth of field and the amount of light that is captured by an image with a given exposure. For an image to be sharp across a large range of depths in the scene, a small aperture is required. However, decreasing the aperture size is not always feasible. Most low-end cameras, such as those found in cellphones, have a fixed aperture size. While the aperture is usually small enough that most of the scene is in focus, photos that require close focusing, such as macro shots, exhibit a shallow depth of field. For cameras that do have control over the size of the aperture, decreasing the aperture size until the entire scene is in focus may not be feasible due to the lack of available light. Small apertures require long shutter speeds, which can result in image blur due to handshake and motion of objects in the scene.

An alternative method for acquiring a single image with a large depth of field is to capture a set of photos focused at different depths, also known as a focal stack (Figure 1). This allows for using larger apertures and shorter exposure times, but each photo will have some areas that are sharp and some that are blurry. The images are then combined to produce a composite that is in focus everywhere using an all-in-focus image fusion algorithm [1].

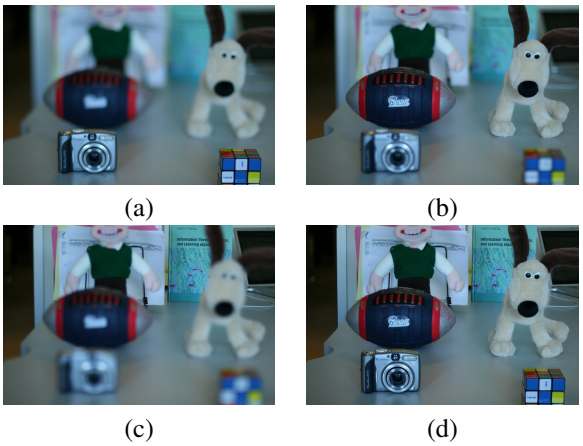


Fig. 1. A focal stack (a–c) is captured by focusing at different distances. The images are then fused to obtain an all-in-focus result (d).

A large body of work in graphics, vision, and image processing has addressed the problem of *combining* focal stacks into all-in-focus images. In contrast, the problem of how to *capture* focal stacks efficiently has received less attention. Hasinoff et al. [2] cover the entire range of depths from the closest focus distance to infinity with the minimum number of shots, given the characteristics of the particular camera. However, the number of images required also depends on the scene: scenes with large depth variations require more images, while scenes where all objects are close to each other require fewer images. We address the efficient *capture* of a focal stack by analyzing both the properties of the camera and the scene geometry.

II. Efficient Capture of Focal Stacks

Why is capturing the smallest number of images important? After all, if the entire depth range is captured, the final result will still have all objects in focus, even if some images are focused on depths that do not contain anything in the scene and are therefore redundant. We are interested in both capturing and processing all the images online on programmable cameras that typically have less processing power and memory than desktop computers. Capturing only as many images as needed for a given scene has several advantages: shorter total capture time, which improves the user experience and has less risk of motion due to handshake and objects in the scene; less

total data that needs to be captured (a dense focal stack may be too large to fit into the main memory of a programmable camera or a camera phone); shorter processing time (all-in-focus algorithms are computationally expensive); and processing more images than needed increases the likelihood of stitching artifacts, potentially lowering the quality of the result.

We have created an all-in-focus capture and processing system that adapts to scene geometry. Our approach is inspired by passive autofocus techniques in digital photography, which, assuming a fixed aperture size, a static camera, and a static scene, analyze a stream of preview images of the scene while the lens sweeps from near to far focus. They determine the plane where to focus to give the overall sharpest image. In contrast, our approach automatically selects a minimal set of images, focused at different depths, such that each part of the scene is in focus in at least one of the images.

Our technique, which is presented in detail in [3], runs in real-time, and its main steps are: (i) continuously varying the lens focal distance through the entire depth range while computing low-resolution (16×12) sharpness estimates, based on a $[-1\ 2\ -1]$ contrast filter efficiently computed by the camera's hardware; (ii) given the sharpness estimates, calculating where the images in the minimal set need to be focused using a novel plane-sweep algorithm; (iii) capturing the resulting minimal set of images in high resolution; (iv) correcting for magnification differences between the images; (v) aligning the images to compensate for handshake [4]; (vi) merging the obtained stack into an all-in-focus result directly on the camera [5]. We proposed a lens-sweep strategy for capturing preview images that optimizes for speed while guaranteeing coverage of the entire depth range.

III. Experimental Results

We apply our technique in the mobile computational photography scenario on a Nokia N900 smartphone. The N900 runs the Linux-based Maemo operating system, and has a 5 MP camera, 600 MHz OMAP 3 processor and 256 MB of RAM. We believe that our implementation is the first fully functional all-in-focus system running entirely on a mobile device.

We used our system to test the algorithm in a few scenes with different variations in depth, by capturing images using a handheld N900. We compared the speed and visual quality of the results from three capture strategies: our method, a full focal stack of 24 images comprising the maximal and disjoint depth of field intervals as to cover the full depth range from 5 cm to infinity, and a standard autofocus approach that results in a single selected plane. The experiments demonstrate that our scene-adaptive method results in faster capture and processing times than the techniques that capture and merge images that cover the entire depth of field independently of the scene. It also minimizes problems with artifacts due to motion, and allows for processing focal stacks directly on mobile devices with limited memory.

Fig. 2 displays the results obtained in one of the scenes, which consisted of three depth layers. Fig. 2(a-c) shows the three images in the focal stack, after magnification correction and alignment. The cropped areas shown below each image illustrate the different levels of blur in different regions. Fig. 2(e) shows the fused result for our technique. Fig. 2(d) shows the fused result for a full focal stack. Fig. 2(f) displays the image obtained by simply running a standard autofocus algorithm and capturing a single image.

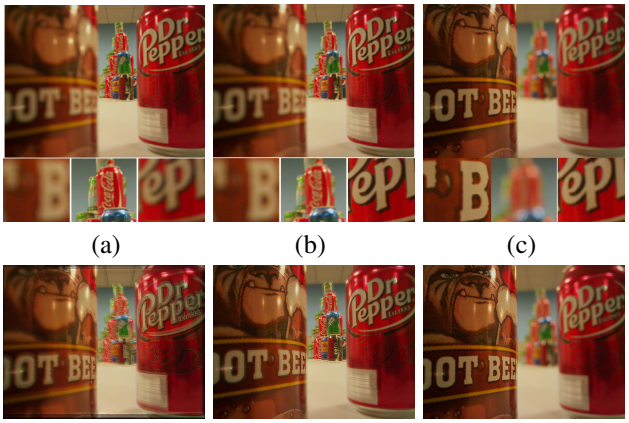


Fig. 2. Scene with three depth layers: (a-c) the input images focused at different distances, (d) full focal stack, (e) our method, and (f) standard autofocus.

As expected, only one layer appears focused in the single image result (Fig. 2(f)). Using the full focal stack (Fig. 2(d)) resulted in several artifacts due to camera motion and accentuated parallax for close objects, which could not be corrected by the alignment algorithm. On the other hand, image alignment works better on the set of 3 images given by our technique, resulting in fewer artifacts (Fig. 2(e)).

References

[1] A. Agarwala, M. Dontcheva, M. Agrawala, S. Druker, A. Colburn, B. Curless, D. Salesin, and M. Cohen, "Interactive digital photomontage," in *Proc. SIGGRAPH*, 2004.

[2] S. W. Hasinoff, K. N. Kutulakos, F. Durand, and W. T. Freeman, "Time-constrained photography," in *Proc. ICCV*, 2009.

[3] D. Vaquero, N. Gelfand, M. Tico, K. Pulli, and M. Turk, "Anonymous submission," 2010.

[4] M. Tico and K. Pulli, "Low-light imaging solutions for mobile devices," in *Proc. of Asilomar Conference on Signals, Systems, and Computers*, 2009.

[5] A. Mihal *et al.*, "Enblend 4.0," http://enblend.sourceforge.net.

A Case for Smartphone Reuse to Augment Elementary School Education

Xun Li Pablo J. Ortiz Jeffrey Browne Diana Franklin John Y. Oliver*
Roland Geyer‡ Yuanyuan Zhou† Frederic T. Chong
Department of Computer Science, University of California, Santa Barbara
{xun, portiz, jbrowne, franklin, chong}@cs.ucsb.edu
† Department of Computer Science and Engineering, University of California, San Diego
yyzhou@cs.ucsd.edu
* Department of Electrical Engineering, Cal Poly San Luis Obispo
jyoliver@calpoly.edu
‡ Donald Bren School of Environmental Science and Management, University of California, Santa Barbara
geyer@bren.ucsb.edu

I. Introduction

Today, personal computing is shifting from traditional desktop computers to mobile devices, such as MP3 players, PDAs and cellular phones. Companies like Nokia, Samsung, Google and Apple have spurred innovation such that today, smartphones come standard with touch screens, GPS, cameras, as well as a whole suite of connectivity options. Moreover, mobile devices will continue to advance in the coming years. The capabilities that seem high-end today will be standard fare in the next generation of devices and basic requirements in even later devices.

However, the dark side of Moore's Law is the waste caused by our society's insatiable desire to frequently upgrade our mobile devices. The typical cellular handset is discarded after only 18 months. The environmental impact of this stream of handsets in terms of manufacturing energy, materials and disposal costs is alarming. Our initial work [2] investigates reusing embedded microprocessors, showing that the energy required to manufacture a processor far out-strips the energy consumed during the processor's lifetime for most low-power embedded processors. Similar relationship also holds for entire smartphone systems. By repurposing retired mobile devices we can significantly amortize production energy of these devices and reduce their environmental impact.

Even if we believe that it is more environmentally friendly and feasible to reuse mobile phones, we need to find a market for these mobile phones. We argue that use in educational classrooms is a promising candidate for reused smartphones, given the potential benefits to elementary school students who may not commonly own cellular phones (or are forbidden from having them in school). Note that we focus on repurposing smartphones in *non-phone* applications and expect cellular calling capabilities to be disabled. Phone manufacturers prefer this model, as it avoids competition with new product lines and avoids the political pitfalls of "dumping" old technologies into "second-class" markets.

Compared with conventional ways of teaching in which students often learn in a passive manner, *education using mobile phones* enables more active models of learning. For example, participatory sensing [4] uses mobile devices to form interactive, participatory sensor networks that enable every user to gather, analyze and share local knowledge.

Unfortunately, we are still far away from reusing smart devices for education purpose due to the fact that the power consumption and resource usage of educational applications on mobile platforms has not been explored, so there are no observations indicating that the resource requirement of educational applications can be satisfied by recycled cellular phones. As a pioneering work for designing life-cycle aware mobile devices, we aim at solving the above problems by making connections between the resource provided by mobile devices and demanded by educational applications, and proposing potential solutions for existing challenges.

II. Educational Application Characterization

To characterize the resource requirement of educational applications on smartphones, we perform different experiments by running the most popular free applications related to elementary school education from the Educational Category of Google ADC Top 200 applications [3] on The HTC Nexus One. We study the static resource requirements for each application including their functional requirements and storage requirements, as well as their dynamic resource usage such as memory usage, power consumption and network communications.

- **Function Requirement** One of the main issues of reusing mobile devices is that phones wear out with long-term use, rendering some components useless. It is critical to investigate the function requirements of educational applications before deciding the feasibility of reusing smartphones for such purpose. We find out that most of the applications (10/12) rely on the network connection, while other functions are less widely used by certain categories of applications (e.g. only 3 applications use the camera). More importantly, we observe that most

applications only need around 3 in all 10 common device features (at most 4), so that damage to any single component will not affect the use of most other applications. Thus, although mobile devices will wear out during use, they can still be reused in most scenarios as long as the network adapter remains functional along with at least a few other components.

- **Storage Requirement** The storage requirement of each application includes the application program and the data. We observe that for each application, the application size dominates the total storage, averaging around 1.2MB, while the data size increases slowly. Assuming a total size of application and data to be 2.6MB, which is the maximum size among the applications, a 190MB flash memory can support installing more than 70 applications. We also calculate that a 512MB flash memory will have a useful lifespan of 6.5 years. According to such degrading speed, a reused smartphone with original internal storage of 512MB after 18 months will still have 400MB functional cells. Taking away the storage requirement by the Android OS (512MB - 190MB = 322MB), 80MB storage space is still available for the applications. Hence a reused Nexus One is still capable of installing more than 30 applications without any difficulty.
- **Memory Usage** We measure that the average memory usage of educational applications is around 8KB, and the peak memory requirement can reach 22KB. This memory requirement is extremely small compared to the available memory space provided by the RAM of the Nexus One; the total capacity of the RAM is 512MB, of which around 300MB is used for system and default background applications including mails and browsers. Around 200MB of memory is still available for running userspace applications.
- **Power Consumption** We measure the real power consumption of running educational application on The Nexus One. The accumulated current is measured by a DC meter interposed between the battery and the phone, while the voltage is sampled by the logger application through operating system interface. From our results we observe that the gap between the power consumption of stand-by and active mode is large: $72mW$ versus $910mW$ as an average when running educational applications. The major difference comes from the power consumption by the touchscreen.

  To understand the battery usage of the smartphone, we obtain a linear relationship between the capacity of the battery and the number of charging cycles for a 1400-mAh mobile battery from [1]. Based on the relation and the assumption that a new battery with 1400mAH capacity contains 5880mWH of energy and is able to last for 32 hours without recharging in a typical usage model [5], we calculate that in general situations a battery could be reused for another 2.3 years after 18 months of regular use.
- **Network Communications** We measure the network communication statistics of educational applications through Android OS interface. The average network requirement for all applications is 655bytes/second in average, and the peak network bandwidth requirement can be as high as 160KB per second. Assuming a big classroom with 100 students, a maximum of 16MB/s bandwidth network is needed. We also calculate that by switching the internet access from the phone network to the Wi-Fi network during using those applications, 12MJ of energy can be saved everyday.

III. CHALLENGING ISSUES IN REUSE

Although the energy cost and environmental impact of manufacturing smartphones has been a pressing problem, and recycled phones are capable of running different types of applications in terms of functions, storages, memories and powers supplies, the heterogeneity of different models of devices holds back potential smartphone reuse. In addition to the heterogeneity generated by vendors, software developers need to consider the following design characteristics to effectively use recycled devices.

- Significantly degraded reliability. Compared to new devices, recycled ones are much less reliable due to wear out.
- Partially configured devices. Some devices may not have all the necessary configurations to run every application.
- Different processing power and timing guarantees. Some devices may have significantly slower processing power and memory/IO access latencies.

IV. CONCLUSION AND FUTURE WORK

In this paper we demonstrate the potential benefits of reusing smartphones by analyzing their manufacturing and life-time energy. Based on our experiments on The HTC Nexus One platform, we show that although different components in smartphones degrade from use, their functionalities, available resources and power supplies are still able to satisfy the requirement of educational applications. Though we use the Nexus One as our unique experimental target, our contributions and observations could be easily extended to other mobile architectures.

Future work will focus on ways of handling the different challenges we identified in this paper. The software development methodology needs to be capable of taking life-cycle issues into account. Moreover, new types of collaborative applications for classroom settings will also be a interesting future work.

REFERENCES

[1] S.S. Choi and H.S. Lim. Factors that affect cycle-life and possible degradation mechanisms of a li-ion cell based on licoo2, 2002.

[2] R Geyer, J Oliver, R Amirtarajah, V Akella, and FT Chong. Life cycle aware computing: Reusing silicon technology. *IEEE Computer*, 40(12):56–61, 2007.

[3] Google Inc. ADC 2 top 200 gallery: Education/reference. http://code.google.com/android/adc/.

[4] Participatory Sensing. J. burke, d. estrin, m. hansen, a. parker, n. ramanathan, s. reddy, m. b. srivastava. *WSW06 at SenSys 06,*, 2006.

[5] Jeff Sharkey. Coding for life–battery life, that is. In *Google I/O*, 2009.

Fighting Fire with Fire: Superlattice Cooling of Silicon Hotspots to Reduce Global Cooling Requirements

Susmit Biswas⋆, Mohit Tiwari⋆, Timothy Sherwood⋆, Luke Theogarajan†, Frederic T. Chong⋆
Department of Computer Science⋆,Department of Electrical and Computer Engineering†
{susmit, tiwari, sherwood, chong}@cs.ucsb.edu⋆, ltheogar@ece.ucsb.edu†

I. Introduction

The running costs of data centers are dominated by the need to dissipate heat generated by thousands of server machines. Higher temperatures are undesirable as they lead to premature silicon wear-out; in fact, mean time to failure has been shown to decrease exponentially with temperature (Black's law [2]). Although other server components also generate heat, microprocessors still dominate in most server configurations and are also the most vulnerable to wear-out as the feature sizes shrink. Even as processor complexity and technology scaling have increased the *average* energy density inside a processor to maximally tolerable levels, modern microprocessors make extensive use of hardware structures such as the load-store queue and other CAM-based units, and the *peak* temperatures on chip can be much worse than even the average temperature of the chip. In recent studies, it has been shown that *hot-spots* inside a processor can generate $\sim 800W/cm^2$ heat flux whereas the average heat flux is only $10 - 50W/cm^2$, and due to this disparity in heat generation, the temperature in hot spots may be up to $30\,^{\circ}\mathrm{C}$ more than average chip temperature.

The key problem processor hot-spots create is that in order to prevent some critical hardware structures from wearing out faster, the air conditioners in a data center have to be provisioned for worst case requirements. Worse yet, air conditioner efficiencies decrease exponentially as the desired ambient temperature decreases relative to the air outside. As a result, the global cooling costs in data centers, which nearly equals the IT equipment power consumption, are directly correlated with the maximum hot spot temperatures of processors, and there is a distinct requirement for a cooling technique to mitigate hot-spots selectively so that the global air conditioners can operate at higher, more efficient, temperatures.

We observe that localized cooling via superlattice microrefrigeration presents exactly this opportunity whereby hot-spots can be cooled selectively and allow global coolers to operate at much more efficient temperatures. Recent advances in processor cooling technologies have demonstrated that thermoelectric coolers (TEC), which use a Peltier effect to form heat pumps, can be used to reduce the temperature of hot spots. By applying a thermoelectric cooler between the heat spreader and the processor die and applying current selectively at the hot spots, heat from the hot-spots can be spread much more efficiently. The ability to implement such thermoelectric coolers on a real silicon device has been demonstrated recently [3], albeit for small prototype chips. The key question then, that needs to be answered before such thermoelectric coolers can be integrated in commodity server processors, is "What is the potential for superlattice microrefrigeration to reduce global cooling costs in data centers?".

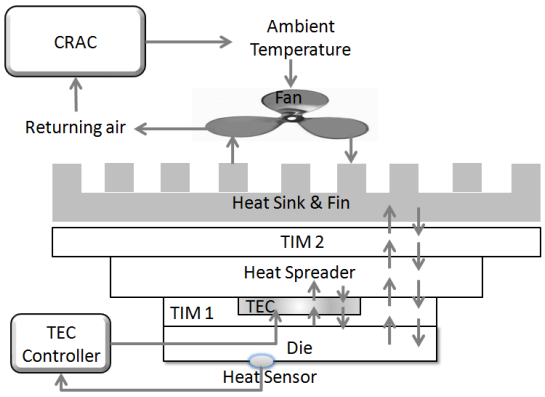


Fig. 1. Heat Flow

In order to answer this question, we present a comprehensive analysis of the impact of thermoelectric coolers on global cooling costs. Our thermal analysis covers all aspects of cooling a server in a data center, and integrates on-chip dynamic and leakage power sources with a detailed heat diffusion model of a processor (that models the silicon to the thermoelectric cooler to the heat spreader and the heat sink) and finally the computer room air conditioner (CRAC) efficiency, as shown in Figure 1. In Section II, we present the components of the system model.

II. Modeling Chip to Data-center Cooling

In order to quantify the benefits of using TEC based cooling, we need to estimate the peak temperature, which determines the mean time to failure (MTTF), as well as the consumption of power by the microprocessor - taking into account the leakage and temperature feedback loop. We use curve fitting techniques along with ITRS [1] data to model the leakage power consumption in the active layer. We use the finite

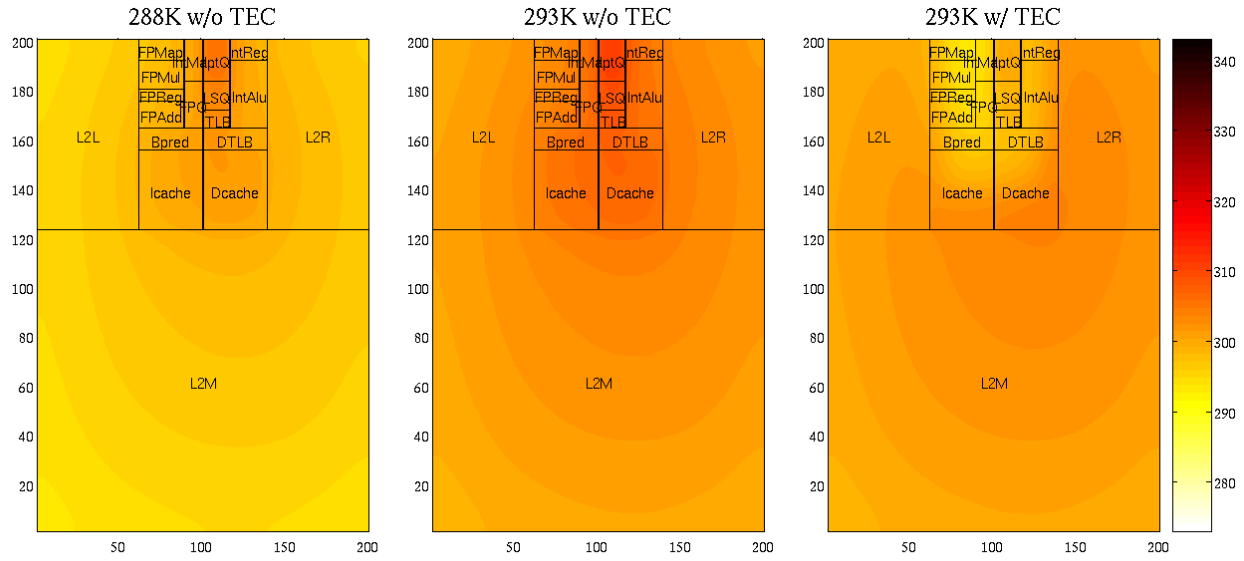


Fig. 2. Effect of TEC of Thermal-profile

difference method of computing the steady state temperature by solving heat diffusion equation (1) with boundary condition of convective cooling (equation (2)) using an explicit method which promises greater scalability for fine grain analysis. TEC is modeled as a heat soure at the hot side and a sink at the cold side with pumping rate as shown in equation (3), and we model CRAC efficiency (Coefficient of Performance) with equation (3) as proposed by Moore *et al.* [4] for estimating the cooling power consumption.

$$T^{t+1} = T^t + \left(\frac{k\Delta t}{\rho C_p}\right)\left[\frac{1}{\Delta x^2}\left(M_x + M_y + M_z\right)T^t\right] + \frac{g\Delta t}{\rho C_p} \quad (1)$$

$$M_x(T_{x,y,z}) = T_{x-1,y,z} + T_{x+1,y,z} - 2.T_{x+1,y,z}(M_y, M_z similar)$$

$$T_{surf+\Delta x} = T_{surf} - \Delta x.\frac{h}{k}.(T_{surf} - T_{amb}) \quad (2)$$

$$g_{TEC} = \frac{1}{\Delta x^2}\left(0.5\frac{\alpha^2}{\rho_e}T_{cold-side}^2\right) \quad (3)$$

$$COP(T) = a_2T^2 + a_1T + a_0 \quad (4)$$

$$P_{cooling} = (P_{leak+dyn}/COP_{CRAC,T_{amb}+\Delta}) + Capacity_{TEC} * Area_{TEC}/COP_{TEC} \quad (5)$$

By integrating all the models, cooling power at $(T_{Amb} + \Delta)$ ambient temperature is estimated as equation (4). We perform our experiments by collecting dynamic power density traces of several SPEC CPU2000 benchmarks. Leakage power is modeled using ITRS data [1] and as a 3^{rd} order function of temperature. We developed a tool that solved the heat diffusion equations (1) - (3) using the finite difference method to estimate the total power consumption and the temperature profile of the chip-package. We present the results only for a representative benchmark *apsi* in Section III.

III. Results

By increasing the ambient temperature by $5\,^{\circ}\mathrm{C}$ ($288K$ to $293K$), the peak temperature of the die increases, and by using a TEC layer, the peak temperature is reduced as shown in Figure 2. We experiment with various TEC pumping

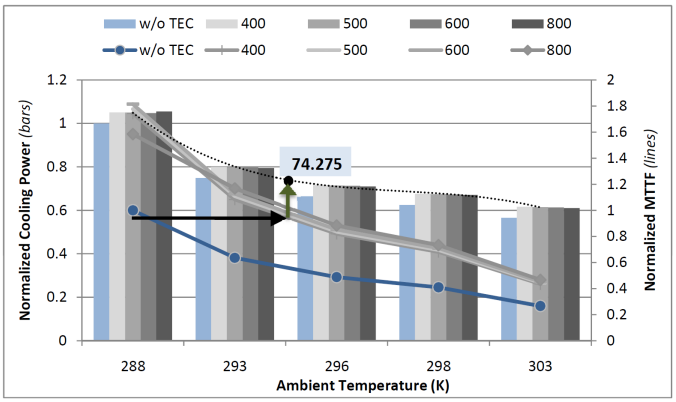


Fig. 3. Effect of TEC on cooling power and MTTF

capacities $(400, 500, 600, 800 W/cm^2)$ and present the result for *apsi* in Figure 3 where bars plotted agains $y1$ axis indicate the cooling power consumption, and MTTF of the hottest point is plotted against the $y2$ axis. Through curve-fitting, we find that TECs allow the CRAC to operate at $6.5\,^{\circ}\mathrm{C}$ higher temperatures $(294.5K)$ which translates into a 25.73% reduction in cooling power, while maintaining the same worst-case on-chip temperatures i.e. without degrading the lifetime of the processor.

References

[1] The International Technology Roadmap for Semiconductors. http://www.itrs.net/.

[2] J. Black. ElectromigrationA Brief Survey and Some Recent Results. *IEEE Transactions on Electron Devices*, 16(4):338–347, 1969.

[3] I. Chowdhury, R. Prasher, K. Lofgreen, G. Chrysler, S. Narasimhan, R. Mahajan, D. Koester, R. Alley, and R. Venkatasubramanian. On-chip Cooling by Superlattice-based Thin-film Thermoelectrics. *Nature Nanotechnology*, 2009.

[4] J. Moore, J. Chase, P. Ranganathan, and R. Sharma. Making Scheduling "Cool": Temperature-Aware Workload Placement in Data Centers. In *ATEC '05: Proceedings of the annual conference on USENIX Annual Technical Conference*, pages 61–75, Berkeley, CA, USA, 2005. USENIX Association.

Information Flow Secure Architectures

Mohit Tivari, Xun Li, Hassan M. G. Wassel, Frederic T. Chong, Timothy Sherwood
Department of Computer Science
University of California, Santa Barbara
{tiwari,xun,hwassel,chong,sherwood}@cs.ucsb.edu

Systems responsible for controlling aircraft, protecting the master secret keys for a bank, or regulating access to extremely sensitive commercial or military information, all demand a level of assurance far beyond the norm. Creating these systems today is an incredibly expensive operation both in terms of time and in terms of money; even assessing the assurance of the resulting system can cost upwards of $10,000 per line of code [1]. Ideally one can demonstrate strict *non-interference*, which requires showing that *sensitive* inputs can never have a measurable effect on an output marked as *non-sensitive*, a task for which traditional microprocessors are very poorly suited. Microprocessors today contain status bits, exceptions, caches, predictors, bus arbiters, and other behaviors that modify the state of the machine and make it extremely hard to show that *no* internal state affected by sensitive information is visible to other components, either directly through the ISA, or indirectly through the resulting differences in behavior or timing.

We propose, for the first time, an architecture that can execute mixed-trust code and yet can be verified at the gate level to provide provably strong information containment. We introduce Execution Leases, an architectural mechanism that makes all information flows (from the gates up) explicit to the programmer, including timing, covert and implicit flows through control/architectural state. Using this mechanism, programmers can explicitly control all information flows through the machine, and write programs that conform to specific policies such as non-interference even while executing untrusted code, conditional branches, and indirect memory accesses. When compared to the prior architecture based on Gate Level Information Flow Tracking [2], Leases yield programs that can be a 100x faster in some cases, several factors smaller, and far easier to program.

Execution Leases: The basic idea behind a lease is that control of a portion of the machine is given over to an untrusted entity for a fixed amount of time and within a fixed range of addresses. After the lease expires, control is yanked back to the trusted code and any remnants of the untrusted actions are purged from the critical machine state such as the PC. Registers and main memory are not part of the critical machine state and retain their values and their security labels even after a lease expires. The hard part is that leases have to be implemented in such a way that i) enforcement of the lease can *never* by affected by tainted data, ii) the *critical* tainted state (e.g. the PC) can be scrubbed leaving no residue of tainted data behind, and iii) that it is clear through a *gate-level analysis* of the flow of information that properties (i) and (ii) hold (e.g. it does not depend on some property of the software or some semantics of some state to show that (i) and (ii) hold).

Because we want our implementation and analysis of Execution Leases to be demonstrably sound from the gates up, we perform this work under the framework of Gate-Level Information Flow tracking (GLIFT [2]). GLIFT is based on the intuition that all information flows, be they explicit, implicit, or even timing channels, look the same at the level of simple logic gates where weakly defined ISA-level descriptions give way to precise logical functions. The authors build on this idea, define precisely the information flow through a NAND gate, and show how to compose information flow through a combination of NAND gates in order to track all information flows through arbitrary logic in a sound fashion.

GLIFT in and of itself does not ensure that critical or untrusted data do not spread across the whole machine, rather it only ensures that as critical or untrusted data spread across the machine it will always be properly tracked. The challenge is to create an architecture capable of containing the flow of information so tightly that, by looking only at the gate-level implementation it is apparent that it cannot leak, while simultaneously retaining enough flexibility to be efficiently programmed to a variety of uses. In [2], a very simple architecture is used to show that this is indeed possible in principle by removing all branch instructions (replacing them with predicates) and ensuring that there is no possible way for the program counter or memory address to *ever* be effected by any tainted data (i.e. no tainted code or indirect memory accesses). This *severely* restricts the usefulness of the architecture – to even lookup an entry in a simple table of size n takes $\emptyset(n)$ steps instead of the $\emptyset(1)$ that it would with a simple indirect load.

Making Lease Semantics Inherent in Gate-Level Implementation Consider the execution of arbitrary tainted code. This case, where the bits of the actual instruction are tainted, is the most difficult to bound. An untainted function (e.g. some trusted function), wishes to call a tainted function (e.g. some arbitrary code). Instead of a

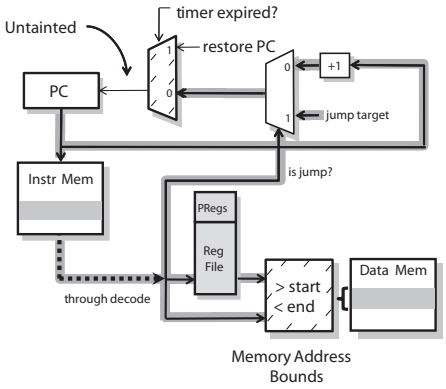


Fig. 1: Execution Lease Architecture: Lease logic (dashed) bounds tainted programs in both time and space, and prevents the entire system state from becoming tainted. PC is restored to an untainted restorePC value when an untainted timer expires. Lease logic is also used to bound the memory regions the tainted code can access.

call-and-return, we can ensure that control will be restored to the leaser context using a timer. In essence, one leases the program counter out to the leasee for a fixed amount of time. Once the timer expires, control is automatically restored back to a return PC value that was provided by the leaser when it invoked the lease. Figure 1 shows the Lease architecture, and a scenario where untainted code leases the CPU to some tainted code. The timer value itself and the restore PC are untainted, and when the timer expires, a MUX is used to reset the PC to the restore PC. Correspondingly, the GLIFT-logic observes that the MUX output is dependent solely on untainted values (i.e the old tainted PC has no effect), and marks the PC as untainted.

Further complexity is introduced by the need to support *multiple nested leases* to support multi-level procedure calls. The need for multiple nested leases naturally suggests maintaining a stack of lease records that stores the time the lease is active for and the PC value that the control must return to when the lease expires. However, following the usual stack semantics, if we always use the current stack pointer to compute the next, once the stack pointer register is tainted, it will prevent itself from ever being reset to untainted. To get around this problem, in our stack implementation, the caller lease sets the stack pointer value to return to when it calls a new lease.

To implement successively nested timers, we encode the timers as a bit-vector where each bit represents a minimum time unit. For instance, a timer of value 00...0111 will execute for three time units. Decrementing these timers then requires shifting the register to the right once every time unit with a 0 entered at the MSB. Nesting of successive timers is enforced at the bit level by using the 0s in the right-shifted current timer value as the prefix of the next timer. This mechanism encodes the timer semantics in its bit-level implementation, and provides gate-level information flow guarantees as opposed to an intuitive scheme that used subtractors to decrement the timers (more details in the full paper).

Similarly, we demonstrate how enforcing memory bounds using a comparator to check whether the address is within bounds leads to the entire memory being marked as tainted. Instead, we encode the memory bounds in a ternary format (where $10**$ represents the range 1000 to 1011) and concatenate the address sent to memory using leading bits from the trusted bound, and replacing the $*$-d don't care bits with the tainted address bits.

ISA, Language and Compiler support for Leases: The flexibility of the Lease architecture is exposed to programmers through two new instructions `settimer` and `setbounds`. The `settimer` instruction allows the caller to specify a fixed time for the callee to execute, and whether the callee can execute general purpose instructions (such as conditional jumps), while the `setbounds` instructions require a power-of-2 aligned range and a timer. We have designed a simple language as a subset of C and adding a `lease (function, mode, time, bounds` statement. A simple compiler is used to help the programmers with deciding the time and memory bounds in GLIFT mode, and performs the job of optimizing the placements of arrays in memory so as to lease them to diffent functions with minimal copying over.

Synthesizable FPGA prototype and Applications: To complete our evaluation of the Lease architecture, we have implemented a CPU prototype in Verilog, augmented it automatically with information flow tracking logic (as in [2]), and synthesized it down to an Altera FPGA. As compared to an Altera-Nios standard processor (1̃000 ALUTs) and the original GLIFT processor in [2] (1̃000 ALUTs), a lease CPU with 8 lease contexts in hardware is 80% larger. With shadow logic, a Lease_sh CPU at 5000 ALUTs grows to 3X as opposed to 1.7X for the original CPU. We believe compiler analysis can be used in the future to bring down the number of hardware lease contexts and bring this overhead down considerably. To demonstrate the improvement in performance, we rewrote all the security kernels in [2]. By introducing protected indirect loads/stores, we see a 68X improvement in AES and 8X in FSM. We also demonstrate the improved programmability bv writing a small encryption library that uses function calls and untrusted I^2C drivers using the Lease language.

References

[1] "What does cc eal6+ mean?" http://www.ok-labs.com/blog/entry/what-does-cc-eal6-mean/.

[2] M. Tiwari, H. Wassel, B. Mazloom, S. Mysore, F. Chong, and T. Sherwood, "Complete information flow tracking from the gates up," in *Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2009.

Internet usage patterns in a rural wireless network in Macha, Zambia

David L. Johnson, Elizabeth M. Belding, Kevin Almeroth
Department of Computer Science, University of California
{davidj, ebelding, almeroth}@cs.ucsb.edu

Gerjan van Stam
Linknet, Macha, Zambia
gertjan.vanstam@machaworks.org

***Abstract*— There have been a number of rural wireless networks providing Internet access over the last decade but little is known about how the Internet is being used, how these networks perform and whether they follow similar trends when compared with Internet usage patterns in developed regions. We analyse a set of network traces from the Linknet wireless network in Zambia, which provides Internet access to approximately 300 residents of a rural village using a satellite link and a combination of point-to-point links, hotspots and wireless mesh networks. Our analysis reveals largely web-based traffic as opposed to the peer-to-peer traffic dominance that one finds in urban areas. Social networking sites receive the most hits, and large file downloads from operating system repositories contribute the most to the bandwidth consumption.**

I. Introduction

There are many rural wireless networks that have been operational since 2005. Some key examples are Airjaldi in India [1], the Peebles Valley Wireless network in South Africa [2] and the Linknet wireless network in Zambia [3]. These networks are unique in that they need to overcome challenges such as long distances between wireless nodes, low-bandwidth gateways to the Internet, lack of reliable power and high cost of Internet connectivity.

Rural wireless networks usually share a low-bandwidth, costly link to the Internet amongst a large user base. This means that any inefficiencies in the network can render a slow shared Internet link almost unusable. Analysing and understanding the traffic distribution, web usage patterns and source of bottlenecks can facilitate network designs that are optimized to give rural users a better Internet experience and bring down usage costs.

There has been a significant amount of work that has tracked Internet usage behaviour in urban areas over the past decade, but there is a large gap in analysis of Internet usage in the small set of rural wireless networks that are now in existence. For example, most recent Internet usage studies show that over half the Internet traffic is peer-to-peer (P2P) traffic. However, P2P traffic over a satellite link is costly and inefficient, and hence is likely to be less prevalent in a rural network.

In this work we analyse a set of Internet gateway trace logs and proxy access logs from the Linknet wireless network in Zambia over a period of 2 weeks in February 2010. The wireless network provides Internet access to approximately 300 residents of a rural village, as well as numerous international visitors. The results show that the network is heavily biased towards social networking-based web traffic and much of the potentially cacheable traffic, like Youtube videos, are not cached.

II. Background and network architecture

Macha, Zambia is a typical poor rural area in Africa with scattered homesteads, very little infrastructure and people living a subsistence lifestyle. However, in the middle of this rural area is a mission hospital, medical research institute and community centre that has provided connectivity to approximately 300 community workers and visitors since 2004 using satellite-based Internet. Over the past few years the Internet connection has been spread to a large portion of staff in the area, as well as the community centre, which has an Internet Cafe. The VSAT connection has a committed download speed of 128 kbps bursting to 1 Mbps and a committed upload speed of 64 kbps bursting to 256 kbps with no monthly maximum. The total monthly cost of the C-band VSAT connection is $1200 (US dollars). The wireless connectivity is spread over 6 square kilometres using standard WiFi clients as well as mesh networks. There are about 100 wireless nodes in the network.

III. Goals and measurement process

Our goals are three-fold: The first goal is to understand the usage patterns of users in a rural context to evaluate differences from an urban setting and gain insight into the needs of users in a rural setting. The next goal is to understand the performance of the network to find out what unique challenges are prevalent in rural networks. Our final goal is to make use of the learning from the first two goals and suggest ways in which the performance can be improved.

IV. Traffic usage analysis

In this section we present an analysis of the usage patterns from squid proxy logs over 14 days. The proxy had a cache hit rate of 43% with an actual bandwidth saving of 19.59%. This low fraction of bandwidth saved is fairly common due to the dynamic nature of the Internet today. The usage followed a typical diurnal pattern but the off-peak period was very short due to users staying up late or waking up early to make full use of the extra available bandwidth during these hours.

Web traffic accounts for 68.45% of the traffic when standard HTTP and secure HTTP are combined and is clearly the dominant protocol. This is in sharp contrast to developed

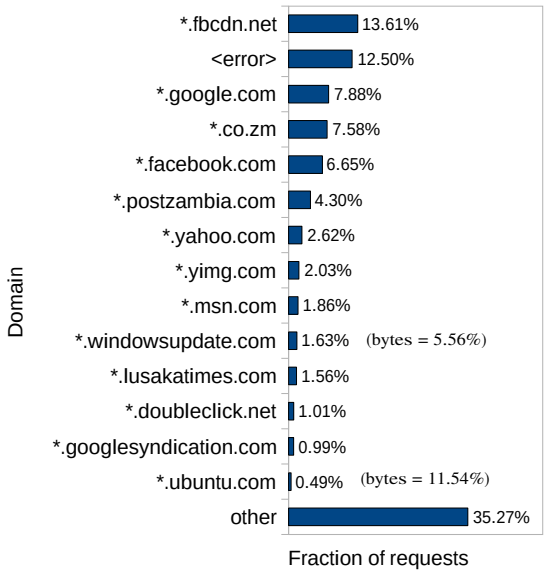


Fig. 1. Web traffic classification by domain visited.

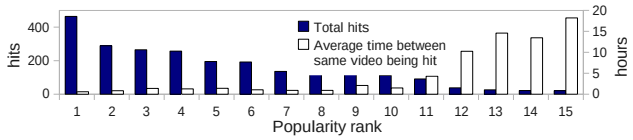


Fig. 2. Youtube popularity.

countries in which 2008/2009 studies show that Web traffic accounts for between 16% and 34% of Internet traffic due to the high prevalence of P2P traffic [4]. 26.47% of traffic could not be classified with simple port based techniques most likely due to traffic like Skype, which is extensively used in Macha, and applications not using known assigned IANA ports.

Figure 1 classifies web traffic into the top 15 site domains. The most startling pattern that emerges is the dominance of Facebook. The Facebook host site and CDN make up 20.26% of the total requests. This is close to 3 times greater than the next most visited site, Google. This presents an amplified correlation with social networking trends that are seen in modern urban networks. Local web sites in Zambia form the third most dominant category, which is encouraging in the sense that local relevant content and language is available to Zambians. There is a large amount of file downloading from package repositories like Ubuntu and Microsoft. 2% of the total web requests are to known advertising domains with more likely in the "other" category. This essentially constitutes wasted bandwidth as the advertising has no relevance to local rural Zambians.

In figure 2, we use our Macha data to rank downloaded Youtube videos by their popularity, and graph the number of hits for the top 15 most popular videos. Of 3162 Youtube hits over two weeks, there were 451 unique videos, but the top 15 ranked videos accounted for 75% of the total Youtube requests. This leads to saturation of the satellite link where downloads fail or are abandoned by the users as no caching is possible for YouTube content using standard squid proxy servers.

V. Recommendations

Due to the limited amount of expert networking skills in the area, a *ClarkConnect* pre-configured gateway server was used in Macha. However, this and other similar pre-configured gateways are not optimized for severely constrained Internet up-links and the following customizations would have a significant positive impact on the network:

- Changing the caching behaviour of squid to be able to identify when the same content is being served by a different URL when accessing content from a CDN like Youtube. This can be done by rewriting the URL request to only preserve the static video ID. Other techniques, such as value-based web caching, deal with dynamic URLs by generating indices based on document content [5].
- Preventing wasted Satellite bandwidth by creating a localization server which uses the social graph in facebook to enable users to share content with each other locally.
- Making use of cache optimizations such as HTTP chunking, time shifting to off-peak hours and data compression to improve the throughput and response times for users.
- Filtering requests to advertising domains such as *doubleclick.net*.
- Making use of a data-ferry synchronization station for repositories like Ubuntu using a dedicated small-footprint PC such as "PC in a plug" that is carried by frequent travellers.

VI. Conclusion

From the results it was apparent that the behaviour of the Internet in a rural wireless network connecting through a satellite is very different from that of an urban network in a developed region. More attention should be given to building pre-packaged networking solutions for rural wireless networks that are cognizant of the characteristics that have been highlighted in this paper. Some of the problems were addressed in the early days of the Internet when last-mile access was similar to what is currently experienced in rural networks, but many of the problems are new as the Internet has become more dynamic and media-rich. As the average web page size continues to grow, the digital divide will widen unless innovative networking techniques, which mitigate these increasing bandwidth demands, are employed.

References

[1] S. Surana, R. Patra, S. Nedevschi, M. Ramos, L. Subramanian, Y. Ben-David, and E. Brewer, "Beyond pilots: Keeping rural wireless networks alive," in *NSDI*, April 2008.
[2] D.L. Johnson, "Evaluation of a single radio rural mesh network in south africa," in *ICTD '07*, December 2007.
[3] J. Backens, G. Mweemba, and G. van Stam, "A Rural Implementation of a 52 Node Mixed Wireless Mesh Network in Macha, Zambia," in *Africomm '09*, December 2009.
[4] H. Schulze and K. Mochalski, "ipoque Internet Study 2008/2009," 2009.
[5] B. Du, M. Demmer, and E. Brewer, "Analysis of WWW traffic in Cambodia and Ghana," in *World Wide Web conference WWW '06*, May 2006.

Fast Nearest Neighbors in Large Networks

Petko Bogdanov, Ambuj K. Singh
University of California, Santa Barbara
{petko, ambuj}@cs.ucsb.edu

***Abstract*—We address the problem of k Nearest Neighbor (kNN) search in networks, according to a random walk based proximity measure. Our approach retrieves the exact top neighbors *at query time* without relying on off-line indexing or summaries of the entire network. This makes it suitable for very large networks, as well as for composite network overlays mixed at query time. We provide scalability and flexibility without compromising the quality of results due to theoretical bound guarantees that we develop and incorporate in our search procedure. We incrementally construct a subgraph of the underlying network, sufficient to obtain the exact top k neighbors. We guide the construction of the relevant subgraph in order to achieve fast refinement of the lower and upper proximity bounds, which in turn enables effective pruning of infeasible candidates.**

We apply our kNN search algorithm on social, information and biological networks and demonstrate the effectiveness and scalability of our approach. For networks in the order of a *million nodes*, our method retrieves the exact top 20 using less than 0.2% of the network edges in a fraction of a second on a conventional desktop machine without prior indexing. When employed for nearest neighbors search in composite network overlays, it scales linearly with the number of networks mixed in the overlay.

I. Introduction

The recent growth of online social and information networks gave rise to the emerging field of *Network Science* [2]. It aims at studying and modeling the behavior of agents, interacting within communication, socio-cognitive and information networks treated as a single composite ecosystem of inter-dependent layers. The individual network layers within this system are typically large-scale and dynamic. Pairwise relations among entities and agents in networks arise from different sources and can be based on multiple features. For example, people may have accounts in several online social networks, targeted towards different interaction types. This diverse social ambiance is complemented by an information layer of email communication, postings and discussion of blog entries or shared photographs. Supporting exploratory and analysis tasks in such composite systems has to be flexible and scalable in order to reflect frequent network changes and user-centric prioritization of the different components.

We propose a scalable approach for k *Nearest Neighbor (kNN)* search in networks. Given a query node in a network, the problem is to identify its k closest nodes. We generalize kNN to multiple networks over the same set of nodes, called network overlays. A scalable kNN search provides an important tool for exploration and analysis of the abundance of networked data. It allows for characterization of the neighborhood of a node and enables diverse applications, such as community identification, anomaly node detection, classification, link prediction and collaborative filtering.

II. Methods

We define a proximity measure *Effective Importance (EI)*, that is related to a recent graph partitioning method [1]. EI captures community structure and exhibits advantageous theoretical properties that allow its efficient bounding with guarantees in an online fashion. EI is defined as the degree-normalized stationary probability of node visit in random walks with restarts (RWR) to the query node. Performing RWR with non-zero restart probability creates a bias, causing neighbors of the restart node to receive more visits per adjacent edge.

We partition the nodes in the graph in three sets: (i) a set of active nodes K; (ii) a set of fringe nodes F, directly connected to nodes in K; and (iii) all other nodes U. The set of known nodes is originally seeded with the restart nodes and its immediate neighbors. Our kNN algorithm operates on the set K, using lower and upper bounds to all network nodes, also computed solely within K. The smaller the active set K, the lower the online search time.

We derive two graph augmentations of the subgraph induced by K that result in lower and upper bounds to the proximity values of all nodes in $K \bigcup F$. In addition we derive an upper bound to all unknown nodes in U. We use our lower and upper bound constructions for pruning nodes that will not be among the k nearest neighbors. Details on the bounds derivation can be found in the technical report [3].

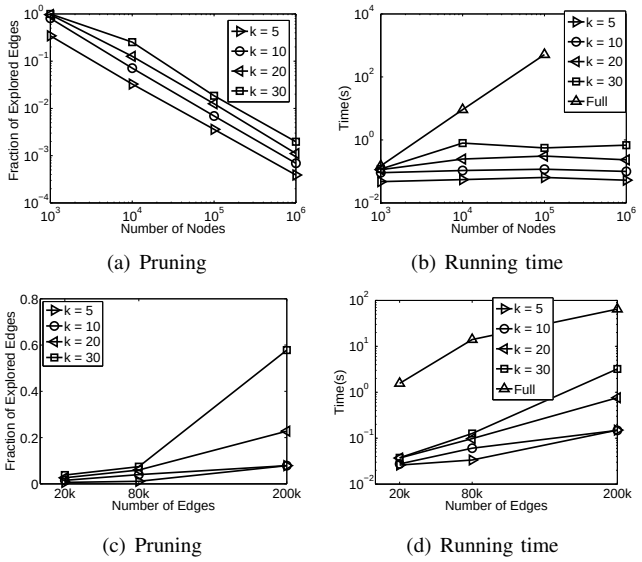
If the bounds are tight for a specific instantiation of the known set K we can return the exact neighbors. When the bound are not tight enough to determine the exact set of top neighbors, we expand K with more graph nodes. Subsequent expansions of K refine the bounds estimation and shrink the node feasibility intervals making guarantees possible for the nearest neighbors set.

Our online kNN search is outlined in Algorithm 1. The input consists of the query node r, the restart probability α, the number of top neighbors k and the network. The set K is initialized with the query node and its immediate neighbors *(line 3)*. Next, we compute the lower and upper bounds *(line 4)* of the EI of nodes in the subgraph $G_{K \bigcup F}$, according to our constructions. The upper bound for all unexplored nodes (part of U) is estimated in *(line 5)*. A series of expansion and refinement steps is performed until the top-k list can be guaranteed using the feasibility intervals of candidate nodes *(lines 6-9)*.

Algorithm 1 Online kNN Search

1: Input: r, α, k and G
2: Output: Ordered set of top-k nodes
3: Initialize $K = \{r\} \bigcup \{j, (i,j) \in E\}$
4: Compute q_{lb}^{α} and q_{ub}^{α}
5: Compute $q_{ub}^{\alpha}(u), u \in U$
6: **while** Top k cannot be guaranteed **do**
7: Extend K with the nodes highest q_{lb}^{α}
8: Refine q_{ub}^{α}, q_{ub}^{α} and $q_{ub}^{\alpha}(u), u \in U$
9: **end while**
10: **return** Top-k nodes

Composite network overlays model the connections of a node in multiple networks in which it participates. We consider kNN search according to user prioritization of the networks in the overlay. In order to select a small relevant subnetwork in a composite network, we push the mixing of the layers into the expansion step of our kNN search (*step 7* of Algorithm 1). We expand with priority-aware best expansion candidates, taking into account feasibility intervals computed according to all layers at the previous expansion iteration.



(a) Pruning (b) Running time

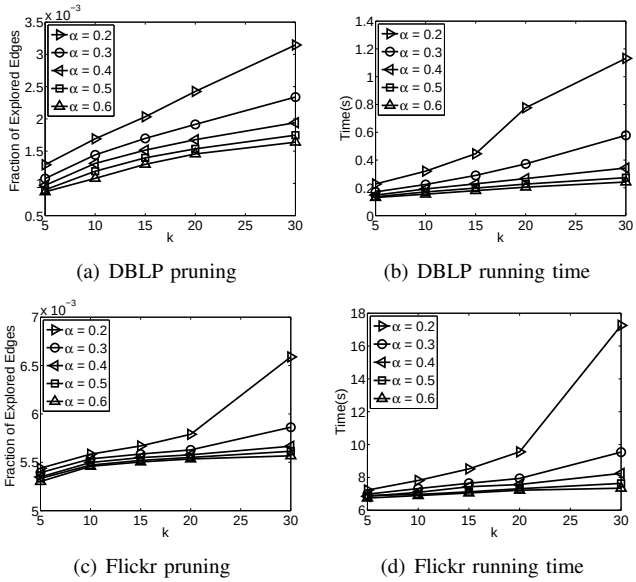
(c) Pruning (d) Running time

Fig. 1. Average performance for increasing number of nodes and fixed average degree of 6 (a), (b); and for increasing number of edges and fixed number of $10k$ nodes (c), (d)($\alpha = 0.3$).

III. Experimental evaluation

We use two real world networks for experimentation. The DBLP co-author network consists of collaboration links between scientific authors based on joint papers. Another large scale network we evaluate contains a *three-million-user* sample of the Flickr social graph. We infer a second Flickr network layer based on common photo favorite bookmarks of users.

We measure kNN's pruning power in terms of the average fraction of edges in the explored network, adjacent to nodes in the active set K. We also report average wall-clock execution time. Details on the datasets, experimental setup and experimentation with composite networks are available in the technical report [3].



(a) DBLP pruning (b) DBLP running time

(c) Flickr pruning (d) Flickr running time

Fig. 2. Average pruning power (a), (c) and online running time (b), (d) for DBLP and Flickr

Figures 1(a) and 1(b) present the scalability of kNN for increasing number of nodes, while keeping the average degree in a synthetic network fixed to 6. The expected growth behavior of scale-free networks is in line with this experiment, since such graphs are typically characterized by a large number of small-degree nodes and a small number of high-degree ones. The size of the kNN-sufficient active subnetwork K remains constant for increasing network sizes. The exact stationary distribution (denoted Full 1(b)) for 100 thousand nodes network takes close to 9 minutes to compute, while our algorithm answers kNN queries for k up to 30 in a tenth of a second, achieving three orders of magnitude improvement.

We evaluate the scalability of our approach for a single synthetic network as it becomes denser (Fig. 1(c), 1(d)). We fix the number of nodes to $10k$ and increase the total number of edges. The average size of the active edge set K is 6% for $80k$ edges, but increases to more than half of all edges when the average degree reaches 20. Note that $10k$ nodes and $200k$ edges corresponds to a dense scenario in which computing the exact EI in the whole network takes $50s$ (trace *Full* in 1(d)).

Our performance evaluation on real-world networks is presented in Fig. 2. We achieve sub-second search time in DBLP for values of k up to 30, while using less than 0.2% of the network edges. The actual EI in the whole network takes more than $150s$ to compute. The top neighbor search for the Flickr friendship graph takes 7 seconds on average, while pruning more than 99.4% of the network edges for $\alpha \geq 0.3$. The increased complexity, compared to the DBLP graph is due to the higher density and bigger size of Flickr.

References

[1] R. Andersen, F. Chung, and K. Lang. Local Graph Partitioning using PageRank Vectors. *FOCS*, pages 475–486, 2006.

[2] A.-L. Barabasi. *Linked: the new science of networks*. Perseus Publishing, 2002.

[3] P. Bogdanov and A. K. Singh. Fast nearest neighbors in large and composite networks. *Technical Report, UC Santa Barbara*, 2010.

Connectors, Mavens, Salesmen and Translators of the Blogosphere

Ceren Budak, Divyakant Agrawal, Amr El Abbadi
Department of Computer Science
University of California, Santa Barbara
{cbudak, agrawal, amr}@cs.ucsb.edu

I. Introduction

Choosing the right set of people to *first* influence on a new idea to maximize the spread of influence in social networks is a computationally expensive problem. Lately there have been various optimization algorithms and models of communication introduced to tackle that problem. However, the scalability issues, the validity of these models and their robustness to small errors in the parameters of the models are unanswered problems that warrant the need to identify simpler methods to serve the same purpose. In a recent work, Gladwell identifies three types of important people that he claims make an idea tip and uses a small number of success cases such as the sudden popularity of the Hush Puppies to support his ideas. We investigate the possible effectiveness of the three heuristics introduced in [2], as well as another heuristic that we call *translators* in a much larger scale.

II. The Law of the Few

A. Preliminaries

A social network can be modeled as a directed graph $G = (N, E)$ consisting of nodes *N* and edges *E*. The historical data H of the set of cascades in G is represented as a set $H = \{c_1, c_2, ..., c_m\}$ where c_i is an ordered list of nodes n_j s.t. $n_j \in N$ where the nodes are ordered w.r.t. the time they *adopt* or *advocate* cascade c_i. We denote the first time n_j appears on the list of c_i as the time it *adopted* cascade c_i and all the occurrences of n_j in c_i as the times it *advocates* cascade c_i. We also denote the time n_j *adopts* c_i as $t_{i,j}$ and this refers to the index of first occurrence of n_j in c_i.

B. Mavens, Connectors, Salesmen and Translators

1) Connectors: *Connector*, translated into a graph, is a node that has high degree centrality. W.l.o.g. let $\langle n_1, n_2, n_3, ..., n_n \rangle$ denote the list of all the nodes in N ordered by the number of their outgoing edges. The top-k Connectors list CON_k consists of nodes $\langle n_1, n_2, n_3, ..., n_k \rangle$.

2) Mavens: The word "maven" comes from Yiddish and means one who accumulates knowledge. According to Gladwell *mavens*: 1) seek new knowledge, 2) cannot help but help others and thus share the knowledge they acquire and 3) an individual is very likely to believe the correctness and importance of this piece of information provided by a *maven*. Translating those features into graph theory, we define *Mavens* as those that start a large number of cascades and have high influence on their neighbors.

In order to locate mavens in a graph G, we first define *influence* formally. $Inf_{i,j}$, influence of n_i on n_j is the ratio of the number of cascades n_i successfully activated n_j to all the cascades n_i tried activating n_j. $Inf_{i,j} = \frac{Success_{i,j}}{Success_{i,j}+Fail_{i,j}}$ where $Success_{i,j} = |\{c_k | n_i \in c_k \wedge n_j \in c_k \wedge t_{k,i} \leq t_{k,j}\}|$ and $Fail_{i,j} = |\{c_k | n_i \in c_k \wedge n_j \notin c_k\}|$. Given this definition of *node-to-node influence*, the influence set of a node n_i can be defined as $InfSet_i = \{n_j | e_{i,j} \in E \wedge Inf_{i,j} != \perp\}$ and the aggregate influence of a node n_i as: $Inf(n_i) = \sum_{n_j \in InfSet_i} Inf_{i,j} / |\{InfSet_i\}|$ W.l.o.g. let $CM = \langle n_1, n_2, n_3, ..., n_n \rangle$ denote N ordered by this measure. We define the set *Candidate Mavens* as the first $\lceil n/k \rceil$ nodes in list CM, i.e. the top $100/k$-percentile *local influentials*.

We then further filter the list CM to only include those nodes that are *original sources* of information. For a cascade c_i, let n_j denote the first blog in the list c_i. We call n_j the *creator* of cascade c_i. The *maven score* of a node n_j can be computed as: $MS(n_j) = |\{c_k | t_{k,j} = 1\}|$ W.l.o.g. let the list CM re-sorted using this score consist of nodes $\langle n_1\prime, n_2\prime, n_3\prime, ..., n_{\lceil n/k \rceil}\prime \rangle$. The top-k Mavens list M_k consists of nodes $\langle n_1\prime, n_2\prime, n_3\prime, ..., n_k\prime \rangle$.

3) Salesmen: A *salesman* is a person with high charisma who can sell ideas to almost anyone since he never gives up [2]. We capture the notion of a *salesman* as having a large number of trials to activate one's neighbors per cascade. We compute the *salesman score* of each node as: $SalesScore(n_i) = \sum_{c_k, s.t. n_i \in c_k} SalesScore_{i,k} / |\{c_k | n_i \in c_k\}|$ where $SalesScore_{i,k}$ is defined as the number of times n_i appears in list c_k. W.l.o.g. let $\langle n_1, n_2, n_3, ..., n_n \rangle$ denote N ordered by $SalesScore(.)$. The top-k Salesmen list S_k consists of nodes $\langle n_1, n_2, n_3, ..., n_k \rangle$.

4) Translators: We also study another class of actors we call *translators*, those that act as bridges among different communities and therefore have the power of changing the context in which to present an idea. We use an overlapping community detection algorithm [1] to discover the communities for which we then seek to find the translators. We believe static properties such as the sole existence of links is not enough to define what makes a community and claim that existence of flow of influence between nodes is a better indication of community and therefore use the cascade history H to detect communities. The nodes that are a part of the same

cascade influence each other and the fact that they belong to the same cascade also indicates similar interests.

The community detection algorithm used [1] consists of two parts: an initialization phase which creates seed clusters; and an improvement phase which repeatedly scans the nodes in order to improve the current clusters until reaching a locally optimal collection of clusters. The *density metric* we use is based on the co-occurrences of nodes in cascades. To this end, we construct a hashtable T to keep track of the co-occurrences of nodes in cascades. The keys in T are of the form (n_i, n_j). Let $V_{T,i,j}$ denote the value of the key (n_i, n_j) in T, i.e. the number of cascades that n_i and n_j both *advocated*. The *density metric* was chosen as W_{ai}, called the average influence, which is defined for a set of nodes Set_k as: $W_{ai}(Set_k) = \sum_{n_i \in Set_k} \sum_{n_j \in Set_k} V_{T,i,j} / |Set_k|$ This will assign a high *density* to a set of nodes that occur frequently in the same cascade and will also avoid assigning too many nodes to one set by offsetting the weight by the number of nodes in the cascade. Having discovered communities this way, the next step is to detect the translators between communities. Let the set of communities detected by employing the algorithm presented in [1] and using the *density metric* W_{ai} be $\{Set_1, Set_2, ..., Set_m\}$. We simply define *translator score* of a node as: $TranslatorScore(n_i) = |\{Set_j | n_i \in Set_j\}|$ W.l.o.g. let $\langle n_1, n_2, n_3, ..., n_n \rangle$ denote N ordered by $TranslatorScore(.)$. The top-k Translators list B_k consists of nodes $\langle n_1, n_2, n_3, ..., n_k \rangle$.

C. Law of the Few in the Blogosphere

We used the August-October 2008 Memetracker data that contains timestamped phrase and link information for news media articles and blog posts from different blogs and news websites. The data set consists of 53,744,349 posts and 2,115,449 sources of information. 819,368 of those sources information are blogs. We use methods similar to the ones presented in [3] to extract cascade information from linking behavior in the blogosphere and discover 744,189 cascades. Using the methods formalized in Section II-B, we identify the *mavens, salesmen, connectors* and *translators* of the blogosphere and study their "possible" effect on the effectiveness of cascades. We loosely use the term "possible effect" to refer to high positive correlation that does not guarantee but indicate possible causality. We first analyzed top-10 cascades and observed that almost all started from a *connector* and involved a *salesman* and *translator*.

Although it is temping to suggest, using these findings, that starting a cascade from a *connector* and reaching out to a *translator* or *salesman* will "guarantee" a successful cascade, this would be an overly simplified conclusion. Therefore, we also study the success of cascades that involve the four actors to investigate if these cascades achieve *better than expected* success. Figure 1 shows some of our findings and presents the cumulative distribution function (CDF) of the size of cascades. The X-axis represents the cascade size whereas the Y-axis represents the ratio of cascades that have at least that many blogs in it in log scale. Consider the y values on the

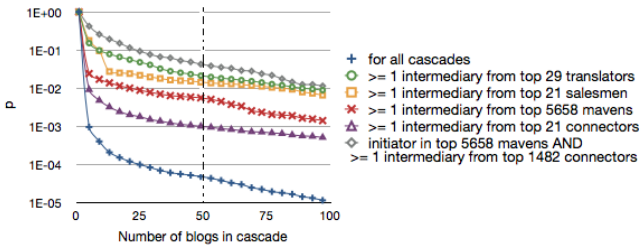


Fig. 1. Comparison of All Actors

dotted vertical line in Figure 1. These values can be used to qualitatively compare how "possibly" effective the actors are in creating cascades of size $>= 50$. Cascades involving any of the four actors have better performance than *expected* (the lowest y value). Cascades that start from a *maven* and have a *connector* as intermediary have the best performance. *Salesmen* and *translators* provide the next best performance, followed by the mavens and connectors respectively. We also compared the success of cascades that start from one of these actors to ones that involve them as intermediaries and observed that the latter is more likely to be several orders of magnitude successful. In a different set of experiments, we seek to answer the question: "Do these actors become part of cascades because they are already successful or do the cascades become more successful because of the involvement of those actors?" Our findings suggest that the latter is the case. We omit the details due to space limitations.

III. Conclusion

In this work, we formally defined four types of actors are shown to have high correlation with success of information cascades in the blogosphere. We also showed that cascades that involve these actors as intermediaries rather than starters are also more likely to be successful and the "possible" effect these actors have in combination is several magnitudes larger than in isolation. These findings can be used to instrument new and scalable algorithms for influence maximization in social networks or to augment current models of communication to capture the observed phenomena.

References

[1] J. Baumes, M. Goldberg, and M. Magdon-Ismail. Efficient identification of overlapping communities. In *ISI*, pages 27–36, 2005.

[2] M. Gladwell. *The Tipping Point: How Little Things Can Make a Big Difference*. Back Bay Books, January 2002.

[3] J. Leskovec, M. McGlohon, C. Faloutsos, N. Glance, and M. Hurst. Cascading behavior in large blog graphs: Patterns and a model. In *SDM*, 2007.

Are BGP Routers Open To Attack? An Experiment

Ludovico Cavedon Christopher Kruegel Giovanni Vigna
Computer Security Lab, Department of Computer Science
University of California, Santa Barbara
{cavedon, chris, vigna}@cs.ucsb.edu

***Abstract*—The BGP protocol is at the core of the routing infrastructure of the Internet. Across years, BGP has proved to be very stable for its purpose. However, there have been some catastrophic incidents in the past, due to relatively simple router misconfigurations. In addition, unused network addresses are being silently stolen for spamming purposes. In this work, we perform a large-scale study to explore the validity of the hypothesis that it is possible to mount attacks against the BGP infrastructure without already having the control of a "trusted" BGP router. Even though we found no real immediate threat, we observed a large number of BGP routers that are available to engage in BGP communication, exposing themselves to potential Denial-of-Service attacks.**

I. Introduction and Related Work

The *Border Gateway Protocol* (BGP, [1]) is the routing protocol at the core of the Internet. BGP is employed to perform routing decision between *Autonomous Systems* (ASes), which are separate administrative domains. Directly connected Autonomous Systems establish *peering* relationships. A peering relationship implies full trust: one router will accept and use for routing any network prefix advertised by its peer routers (unless administratively forbidden).

This full trust between peers is one of the weaknesses of the protocol. In fact, unconditional acceptance and propagation of routing information coming from other peers might render the whole Internet routing stability vulnerable to malicious, compromised, or just misconfigured BGP routers. According to Mahajan et al. [2], BGP misconfiguration is quite common (up to 1% of the global BGP table entries), although only 4% of these misconfigured announcements result in disrupted connectivity. Nevertheless, wrongly advertised prefixes sometimes gave place to well-known catastrophic routing incidents ([3], [4]). In addition, an AS might generate malicious BGP prefix advertisements in order to hijack some IP addresses and use them as not-yet-blacklisted sources of spam ([5], [6]).

Previous literature has been investigating possible areas of attack against the BGP protocol ([7], [8], [9]). Much work has focused on finding solutions to the above-mentioned attacks. For example *S-BGP* ([10]), *soBGP* ([11]), and *IRV* ([12]) are BGP extensions aiming at authenticating prefix origins and updates. However, most of the proposed solutions imply a heavier load on the CPU and memory of routers in order to perform cryptographic operations. Moreover, changes to the protocol need to be backward compatible, as it not possible to replace all the routers software at once. For these reasons, adoption of the defense techniques proposed so far has been extremely slow.

Almost all of the successful attacks against BGP considered in the literature require the attacker to have control of a BGP router. This condition, however, is not easy to achieve and maintain, while having a malicious behavior. However, there is an unanswered question: Is it possible to mount attacks in order to disrupt inter-domain routing without already having the control of a "legitimate" BGP router? In this work, we perform a large-scale study to explore the validity of this hypothesis.

First, we tried to identify how many BGP speakers were reachable on the Internet, by performing a SYN scan for TCP port 179 over a very large part of the Internet address space (about 73%), in order to identify processes listening on that TCP port. Clearly, there is no implication that those hosts were BGP speaker. Restricted to this subset of IP addresses, two additional scans went further in the connection negotiation, aiming at detecting whether the counterpart was willing to continue after a 3-way handshake and even establish a BGP session.

II. Scanning the Internet for BGP routers

Scanning the whole Internet address space is an activity that poses some challenges by itself. The number of IP addresses to probe is about 3.7 billions, which means that in order to complete the scan in 2 weeks, with maximum 1 retry per IP address, we need a constant outgoing packet rate of 6,000 SYN packets per second. Such activity is very noisy, both from the side of the scanner's ISP and the side of the ASes receiving the scan probes. In particular, the port 179 is notoriously a very low traffic port, and, independently from how slowly the scan is performed, a whole network sweep on that port is going to be easily detected. Moreover, scanning activity is usually identified as malicious, and sometimes finds a hostile response from network administrators. For this reason, such a large-scale scanning constitutes a one-shot experiment. We took a number of precautions in order to reduce the impact of our scanning activity on remote networks, and we tried to address in advance potential complaints about our scanning. During our scanning activity we received complaints from 10 different institutions, asking to be excluded from further scans. Only one other institution (AT&T) contacted us letting us know that

they were aware of our activity and that they were interested in hearing about the details of our experiment.

Once we collected the list of IP addresses answering to our SYN packets, we tried to engage in a BGP exchange with them.

III. Results

During our initial TCP SYN scan (carried on between December 2008 and January 2009) about 2.2 million hosts were reported to have the TCP port 179 open (i.e., 0.8‰ of the scanned IP addresses). This pool of potential BGP routers was then probed with our Python BGP speaker in February 2009 and again in February 2010 (Table I). Quite unexpectedly, 35% of these hosts no longer responded to our BGP connection requests. The likely explanation for the phenomenon is the triggering of rate-limiting mechanisms in the target networks. In fact we found that 90% of the timing-out IP addresses were belonging to less than 1% of /16 networks in the scanned address space. Such networks are probably honeypots, employed to monitor malicious activity on the Internet.

After send	*event/response*	*n. of IPs*	*% of probed*	*% of conn.*
SYN	timeout	1026798	47.40	-
	TCP RST	55223	2.55	-
	UNREACH	142257	6.57	-
SYN +ACK	TCP RST	24484	1.13	2.60
	TCP FIN	402381	18.57	42.71
	BGP NOTIF CEASE	8787	0.41	0.93
	non-BGP data	297	0.01	0.03
BGP OPEN	timeout	441130	20.36	46.83
	TCP FIN	51631	2.38	5.48
	BGP NOTIF OPEN	7843	0.36	0.83
	BGP OPEN & UPDATE	5	<0.01	<0.01
	non-BGP data	4994	0.23	0.53

Table I
Results of February 2010 scan. The first column indicates the last packet we sent during our probe, the second column indicates the last packet received from the remote host (or the firing of a timeout in case no data was received). The fourth column indicates the percentage over total probed IP addresses, while in the last column the percentage is computed over the hosts that completed the TCP 3-way handshake.

Of the hosts that completed the TCP 3-way handshake, 45% closed or reset the connection right away. This behavior is probably determined by the interaction between a user space process and the router kernel for the management of the TCP connection. Let us consider the case of a BGP speaker process running in user space. The 3-way handshake is managed by the kernel, which is not aware of the IP addresses of the configured peers (unless some ad-hoc firewall rules have been setup). Therefore the TCP handshake always succeeds and the relative socket is passed to the BGP process. Only at this point the BGP process can decide to terminate the connection.

About 0.4% (7,843) of the hosts concluding the 3-way handshake parsed our *OPEN* message, but declined the BGP peering session with a *NOTIFICATION OPEN Error* message. Almost all these BGP speakers also sent us an *OPEN* message, therefore identifying themselves. They turned out to be actually only 1258 distinct routers (i.e., distinct BGP identifiers), belonging to 318 different ASes. Among them, 3497 routers declared themselves to be in AS numbers reserved for private use, lowering to 192 the number of public ASes for which BGP routers were discovered.

In addition, we found 5 routers (all of them from the same AS) who accepted our BGP peering request and started sending us *UPDATE* messages with BGP prefixes. At this point, theoretically, we should have been able to send prefix *UPDATE*s back to the BGP router. It turned out that we were not actually dealing with some real BGP routers. However, the owner of those BGP speakers confirmed that their security had to be improved and sending them updates could have been a way to overload their database server, where received BGP updates were being stored.

IV. Conclusions

We could not find any real immediate threat. Nevertheless, we found a large number of BGP speakers that would establish a connection with us and possibly accept some input from us. Such routers were not required to engage in such communication, and, in doing so, they exposed themselves to a potential menace, as BGP is extremely sensitive to Denial-of-Service attacks. Moreover, given the amount of trust given to the other peers, a compromised router could seriously affect the whole Internet routing infrastructure. For this reason, it is very important that some equivalent of the principle of *least privilege* is applied also to the acceptance of BGP connections, shielding the routers as much as possible from potentially malicious traffic.

References

[1] Y. Rekhter, T. Li, and S. Hares, "A Border Gateway Protocol 4 (BGP-4)," RFC 4271 (Draft Standard), Internet Engineering Task Force, Jan. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4271.txt

[2] R. Mahajan, D. Wetherall, and T. Anderson, "Understanding BGP misconfiguration," in *SIGCOMM 2002*. ACM, 2002, pp. 3–16.

[3] RIPE NCC, "YouTube Hijacking: A RIPE NCC RIS case study," http://www.ripe.net/news/study-youtube-hijacking.html, 2008.

[4] V. J. Bono, "7007 Explanation and Apology," http://www.merit.edu/mail.archives/nanog/1997-04/msg00444.html, Apr. 1997.

[5] A. Ramachandran and N. Feamster, "Understanding the network-level behavior of spammers," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, p. 302, 2006.

[6] C. McArthur and M. Guirguis, "Stealthy IP Prefix Hijacking: Dont Bite Off More Than You Can Chew," in *Proc. ACM SIGCOMM*, 2008.

[7] O. Nordström and C. Dovrolis, "Beware of BGP attacks," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 1–8, 2004.

[8] K. Butler, T. Farley, P. McDaniel, and J. Rexford, "A survey of BGP security issues and solutions," *AT&T Labs Research*, 2008.

[9] A. Pilosov and T. Kapela, "Stealing The Internet," DefCon 16, 2009.

[10] S. Kent, C. Lynn, and K. Seo, "Design and analysis of the secure border gateway protocol (S-BGP)," *Proc. of DISCEX00*, 2000.

[11] N. James, "Extensions to BGP to support secure origin BGP (sobgp)," *Network Working Group, Cisco Systems*, 2002.

[12] G. Goodell, W. Aiello, T. Griffin, J. Ioannidis, P. McDaniel, and A. Rubin, "Working around BGP: An incremental approach to improving security and accuracy of interdomain routing," in *Proc. NDSS*, vol. 3, 2003.

Hacking for Fun and Education: Organizing the UCSB iCTF

Bryce Boe, Nicholas Childers, Giovanni Vigna
Security Group, Department of Computer Science
University of California, Santa Barbara
{bboe, voltaire, vigna}@cs.ucsb.edu

I. Introduction

Computer security competitions and challenges are a way to foster innovation and educate students in a highly-motivating setting. In recent years, a number of different security competitions were carried out, each with different characteristics, configurations, and goals. From 2003 to 2007, we, the Security Group at UCSB, carried out a number of live security exercises involving dozens of universities from around the world. These exercises were designed as "traditional" Capture The Flag (CTF) competitions, where teams both attacked and defended a virtualized host that provided several vulnerable services. In 2008 and 2009, we introduced two completely new types of competition: a security "treasure hunt" and a botnet-inspired competition. These two competitions represent the largest live security exercises ever attempted and involved hundreds of students across the globe. In this paper, we overview security competitions and detail our 2009 competition with the goal of highlighting their usefulness as a security education tool.

II. Security Competitions

An important aspect of computer security education is hands-on experience. Despite the importance of foundational security classes that focus on more abstract concepts in security, Internet security often requires substantial hands-on training in order to be mastered. Therefore, it is important to improve security training by providing novel approaches that complement the existing educational tools normally used in graduate and undergraduate courses on computer security.

A class of these tools is represented by *security competitions.* In these competitions, a number of teams compete against each other in some security-related challenge. As an educational tool, these competitions have both advantages and disadvantages. A notable advantage is that competition motivates students to go beyond the normal "call of duty" and explore original approaches, sometimes requiring the development of novel tools. Another advantage is that students usually operate against a determined opponent while under strict time constraints and with limited resources thus mimicking a more realistic situation than one can reproduce using paper-and-pencil *Gedanken* experiments. Unfortunately, security competitions have one major disadvantage: they usually require a large amount of resources to design, develop, and run [1], [2].

The best-known online security competition is the DefCon CTF, or Capture the Flag, which is held annually at the DefCon convention. At DefCon 2010, nine teams received an identical copy of a virtualized system containing a number of vulnerable services. Their goal was to secure this set of services whilst concurrently compromising other teams' services allowing them to "capture the flag" and score points [3]. Despite DefCon's nonspecific focus on security education, the competition inspired several editions of the UCSB International Capture The Flag (iCTF). One significant difference between the UCSB iCTF and DefCon's CTF is that the iCTF involves educational institutions spread across the world, whereas the DefCon CTF allows only locally-connected teams. DefCon therefore requires the physical co-location of the contestants thus constraining participation to a limited number of teams.

III. The UCSB iCTF

Annually, since 2003, we organized an international, wide-area security competition involving dozens of teams throughout the world. The goal of these live exercises was to test the participants' security skills in a time-constrained setting. Our competitions were designed as educational tools, and were open to educational institutions only. From 2003 to 2007, each edition of the competition followed the traditional CTF format, where remote teams competed against each other by leveraging both attack and defense techniques. Subsequent editions of the competition grew in the number of teams participating and in the sophistication of the exploitable services. The design, however, remained substantially unchanged, and hence tended to favor iCTF veterans. Therefore, in both 2008 and 2009, we took a different approach in designing the iCTF. We constructed two unique attack-based competitions that mimicked real world scenarios. Here we will focus only on the 2009 competition.

The 2009 iCTF. The 2009 iCTF mimicked the world of malware and accordingly incorporated many features unique to the physiology of modern botnets. In this iCTF edition, each team played the role of an evil botmaster, competing against other botmasters for the control of a large number of simulated users. This iCTF was the largest security competition to date, with 56 teams representing more than 800 participants.

Figure 1 depicts the 2009 game play. Scripts simulated users that were to be compromised and controlled by the participants. Each simulated user followed a cyclical pattern: First the user logged into their *Robabank* account using one of twelve

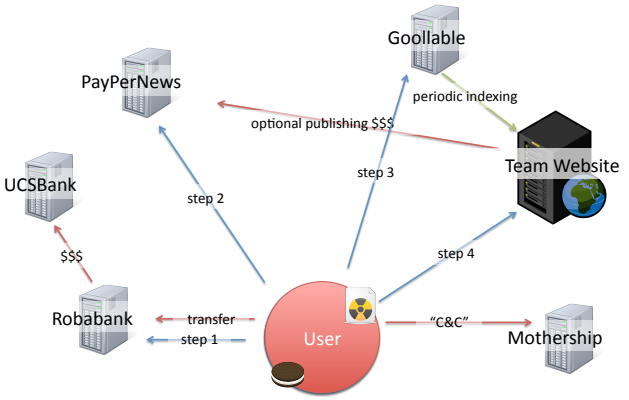


Fig. 1. 2009 competition overview.

vulnerable browsers and consequently retrieved an authentication cookie. The user then visited the *PayPerNews* news site and randomly extracted a keyword from the site's content. Teams had the ability to influence *PayPerNews*'s content by publishing for a fixed cost. At this point, the user searched for the keyword using the *Goollable* search engine. *Goollable* routinely crawled each team's editable webpage thus providing teams with the ability to influence their search ranking through search engine optimization techniques. Finally, one of the links returned by *Goollable* was chosen and visited by the user's browser according to a Pareto distribution. At this point, the browser, and consequently the user, were possibly compromised by a drive-by-download attack delivered by the team's website.

The goal of each participant was to lure a user to a web site under the participant's control, perform a drive-by-download attack against the user's browser, and take complete control of the user. Once the user was compromised, a team had to do two things: i) transfer the money from the user's *Robabank* account to their own *UCSBank* account in order to accumulate "money points," and ii) establish a connection from the user to a remote host, called the *Mothership*, on which to send information identifying the compromising team. A team gained "botnet points" by performing this action. This last step was introduced to generate traffic patterns resembling the interaction of bots with Command-and-Control (C&C) hosts in real botnets. Finally, solving side challenges offered teams a third way to gather points. Challenges varied in type (e.g., binary reversing, trivia, forensics) and difficulty. Teams were awarded "leetness points" for solving a challenge.

This rather complex system of inter-operating components was a central aspect of the 2009 iCTF. That is, instead of just concentrating on single services or single aspects of the game, the participants were forced to understand the *system as a whole*. Even though this aspect generated some frustration with the participants who were used to the straightforward designs of previous competitions, the purpose of the complexity was to educate the students on understanding security as a property of complex systems and not just as a property of single components.

IV. Discussion

Over the years of hosting the iCTF we have learned a few lessons. One in particular is that too much novelty can hurt the overall competition. As opposed to the straightforwardness of a traditional CTF, the 2009 iCTF competition structure was vastly more complicated. Not only did teams have to reverse-engineer the browser software, they also had to perform search engine optimization to get users to visit their sites. Moreover, once they understood how to capture users, they had to figure out how the banking system worked, as well as how the botnet *Mothership* could be used to score points. In total, there were three different kinds of points, with a fairly complex relationship between them that many participants found confusing.

A final thought about security competitions is: *are they worth the effort?* Preparing all the editions of the iCTF took an enormous amount of time and resources. Therefore, it is understandable to wonder what are the benefits. The fact that many similar competitions surfaced after the introduction of the UCSB iCTF shows that there is interest for these competitions. Furthermore, as evidenced by the feedback received following the 2009 competition, our security competitions foster group work and creative thinking, thus it is our firm belief that live exercises are a useful tool to support the security education of students.[1]

References

[1] G. Vigna, "Teaching Hands-On Network Security: Testbeds and Live Exercises," *Journal of Information Warfare*, vol. 3, no. 2, pp. 8–25, 2003.

[2] ——, "Teaching Network Security Through Live Exercises," in *Proceedings of the Third Annual World Conference on Information Security Education (WISE 3)*, C. Irvine and H. Armstrong, Eds. Monterey, CA: Kluwer Academic Publishers, June 2003, pp. 3–18.

[3] C. Cowan, S. Arnold, S. Beattie, C. Wright, and J. Viega, "Defcon Capture the Flag: defending vulnerable code from intense attack," in *Proceedings of the DARPA Information Survivability Conference and Exposition*, April 2003.

[4] N. Childers, B. Boe, L. Cavallaro, L. Cavedon, M. Cova, M. Egele, and G. Vigna, "Organizing large scale hacking competitions," in *Detection of Intrusions and Malware, and Vulnerability Assessment*, ser. Lecture Notes in Computer Science, C. Kreibich and M. Jahnke, Eds. Springer Berlin / Heidelberg, 2010, vol. 6201, pp. 132–152.

[1]For a more in depth discussion of the UCSB iCTF please read our paper, "Organizing Large Scale Hacking Competitions" [4].

DYMO: Linking Network Traffic to Application Code

Bob Gilbert, Richard Kemmerer, Christopher Kruegel, and Giovanni Vigna
Department of Computer Science
University of California, Santa Barbara
{rgilbert, kemm, chris, vigna}@cs.ucsb.edu

I. INTRODUCTION

Most current malware programs rely heavily on the network to carry out their nefarious tasks. For example, malicious software scans for and exploits vulnerable services on remote hosts, sends unsolicited spam mails, leaks data to drop zones, or connects to command and control (C&C) servers for fresh commands and status updates.

Unfortunately, when monitoring the network, it is often difficult to distinguish connections that are established by a malware program from those that are initiated by legitimate applications. One problem is that both legitimate and malicious programs make use of the same application level protocols, such as HTTP or SMTP. Thus, there is often not enough context available at the network level to understand if a connection is desirable. For example, when observing an SMTP connection over which a mail is transmitted, it is difficult to determine whether the user actually composed this mail, or if rather it is spam sent by a bot.

To increase the information available about network connections and to enable better security decisions at the network level, it would be valuable to know which application at the end host is responsible for a certain connection (or packet). That is, one would like to have more context with regard to the origin of each packet on the wire. Such provenance information can be used by network elements (such as routers and firewalls) as well as network services to enforce security policies. For example, a mail server can decide to increase the spam score for mails from unknown mail clients, or a firewall can decide to block all network traffic that is not sent by well-known applications.

In this paper, we propose DYMO, a system that provides provenance information to all outgoing network traffic. More precisely, we introduce a host-based component that marks each TCP connection and UDP packet with an *identity label* that corresponds to the application code that has generated the traffic. Whenever a process opens a network connection, our system injects the identity label of this process into the connection. The label is injected into the option field of the IP header, therefore it can be easily inspected by both network devices and the host that receives the traffic.

We have implemented our system for Windows XP and tested it on several hardware platforms (a "bare metal" installation and two virtualized environments). Our experiments show that labels are the same when the same application is run on different systems. Moreover, when malware attempts to inject code into a legitimate application, the label for this application is correctly updated.

II. DYMO: DESIGN AND IMPLEMENTATION

In this section, we describe the two main components of the system. First, in Section II-A, we discuss how our system tracks the executable regions of a process and uses this information to compute identity labels. Then, in Section II-B, we discuss how labels are injected into outgoing network traffic. DYMO runs entirely in the kernel, monitoring all processes and intercepting all outbound network traffic for label attribution. This architecture allows DYMO to track processes effectively and transparently.

A. Identity Label Generation

An identity label encapsulates all memory regions (sets of consecutive memory pages) of a process' address space that are executed. Since each executable memory region is self contained and can be modified independently, DYMO tracks them individually through *region labels*. While DYMO tracks both image code segments and allocated memory regions marked executable, here we only describe the former.

Region labels are constructed using Huffman coding. A region's variable-length code is retrieved from a precomputed database that maps a cryptographic hash (MD5) of the region to the corresponding code. An identity label is generated by concatenating the individual region labels.

It is easiest to understand the operation of DYMO by walking through the loading and execution of an application. A process is started through an API call (e.g., `CreateProcess`). After the initial thread is created, but before execution begins, DYMO is notified through a process creation callback registered with the operating system. At this point, DYMO constructs a *process profile* to track the process throughout its execution.

Just before the initial thread starts executing, an image loading callback (also registered with the OS) is invoked to notify DYMO that the application's image (the `.exe` file) and the `Ntdll.dll` library have begun loading. DYMO locates the code region (segment) for each of these images in the process' virtual address space and modifies the page protection to remove execute access from the region. DYMO then adds the original protection (`PAGE_EXECUTE_READ`), the new

protection (PAGE_READONLY), the region start address, and the region size to the process profile.

Ntdll.dll is responsible for loading all other required DLL images into the process, so the initial thread is set to execute an initialization routine in the Ntdll.dll code segment. Since DYMO has removed execute access from the Ntdll.dll code segment, the execution attempt raises a DEP/NX [1] exception, which results in a control transfer to the page fault handler. DYMO hooks the page fault handler, so it first gets an opportunity to inspect the fault. DYMO determines that this is the DEP/NX violation that it induced, and it uses the process profile to match the faulting address to the Ntdll.dll code segment. Using the memory region information in the process profile, DYMO retrieves the region label that identifies Ntdll.dll.

The region label is then added to the process profile. Finally, DYMO restores the original page protection (PAGE_EXECUTE_READ) to the faulting region and dismisses the page fault, which allows execution to continue in the Ntdll.dll initialization routine.

Ntdll.dll consults the .exe image file's Import Address Table (IAT) to look for required DLLs to load (and recursively consults these DLLs for imports), and maps them into memory. As before, DYMO is notified of each of these image loads through a callback, and it carries out the same processing described above for each library. The callback is also invoked when DLLs are dynamically loaded during runtime, which enables DYMO to process them as well. After loading, each DLL will attempt to execute its entry point, a DEP/NX exception will be raised, and DYMO will add a region label for each DLL to the process profile.

B. Identity Label Injection

DYMO intercepts outbound network traffic to inject all TCP connections and UDP packets with the identity label of the originating process. DYMO accomplishes this by injecting a custom IP option into the IP header of each packet, which makes it easy for network devices or hosts along the path to analyze the label.

The *injector*, a component that is positioned between the TCP/IP transport driver and the network adapter, does the injection to ensure that all traffic is labeled. A second component, called the *broker*, obtains the appropriate identity label for the injector.

1) The Injector: The injector component is implemented as a Network Driver Interface Specification (NDIS) Intermediate Filter driver [2]. It sits between the TCP/IP Transport Provider (Tcpip.sys) and the network adapter, which allows it to intercept all IP network traffic leaving the host. Due to the NDIS architecture, the injection component executes in an arbitrary thread context. Practically speaking, this means that the injector cannot reliably determine on its own which process is responsible for a particular network packet. To solve this problem, the injector enlists the help of a *broker* component (discussed below).

When a TCP or UDP packet is passed down to the injector, it inspects the packet headers and builds a *connection ID* consisting of the source and destination IP addresses, the source and destination ports, and the protocol. The injector queries the broker with the connection ID and receives back a process identity label. The label is injected into the outbound packet as a custom IP option, the appropriate IP headers are updated (e.g., header length and checksum), and the packet is forwarded down to the network adapter for delivery.

2) The Broker: The broker component assists the injector in obtaining appropriate identity labels. The broker has an interface to receive a connection ID from the injector and maps it to the ascribed process. It then obtains the label associated with the given process and returns the label back to the injector.

The broker is implemented as a Transport Driver Interface (TDI) Filter driver [2]. It resides just above Tcpip.sys in the transport protocol stack and filters the TDI interfaces used to send TCP and UDP packets. Through these interfaces, the broker is notified when a process sends network traffic, and it parses the request for its connection ID. Since the broker executes in the context of the process sending the network traffic, it can maintain a table mapping connection IDs to the corresponding processes. As described above, the injector queries the broker, submitting a connection ID that is resolved into an identity label.

III. CONCLUSIONS

This paper introduces DYMO, a system that labels network packets with information that allows one to determine which code is responsible for the generation of the traffic. The system uses a host-based monitoring component to make sure that the code associated with the execution of an application can be reliably tracked. By associating traffic to application code in a trustworthy fashion, it is possible to enforce a number of application-specific network policies, such as application white-listing or providing different levels of services to different applications. We have developed a prototype of our approach for the Windows XP operating system, and we have evaluated it in a number of realistic settings. The results show that the system is able to reliably track the code base of an application while incurring an acceptable performance overhead.

REFERENCES

[1] Microsoft Corporation, "A detailed description of the Data Execution Prevention (DEP) feature," http://support.microsoft.com/kb/875352.

[2] ——, "WDK and Developer Tools," http://www.microsoft.com/whdc/devtools/wdk/default.mspx.

Quantifying the Environmental Advantages of Large-Scale Computing

Vlasia Anagnostopoulou, Heba Saadeldeen, Frederic T. Chong
Department of Computer Science
University of California, Santa Barbara
{vlasia, heba, chong}@cs.ucsb.edu

I. Abstract

The practical advantages of pay-as-you-go, scalable computing have made large-scale cloud computing services an appealing option for many consumers. At the same time, large-scale datacenters have attracted attention as one of the fastest growing segments of carbon production. In this paper, we attempt to quantify the footprint of various sizes of datacenters in the context of two popular types of small-scale business applications (represented by TPC-C and TPC-H). We evaluate energy, materials and cost as systems scale, accounting for infrastructure, provisioning for future growth, and underutilized resources.

II. Introduction

Large datacenter providers, like Amazon, Microsoft, Google etc, have been quick to create a profitable model under which they rent their machines to clients by the unit, a trend which we have come to know as Cloud Computing. Cloud Computing has offered new possibilities to small companies, especially start-ups, as it allows them to harness substantial processing power and large storage without the cost of deploying and maintaining an actual computing system locally.

Much interest has been directed towards evaluating datacenters, primarily in terms of Total Cost of Ownership (TCO), Energy/USD, and Performance/USD ([1], [2], [3]). The purpose of TCO analysis is typically to constuct or upgrade a datacenter in such a way so that the TCO, consisting of the capital and the operational costs, is minimized. This type of analysis has gone a long way, since these terms are expressed in financial terms well understood by investors. However, the deployment and operation of a datacenter contains significant environmental implications, which are typically ignored from such analyses. For example, Williams [4] pointed out that the manufacturing of a desktop computer is an order of magnitude more intensive than many other manufactured goods, requiring 11 times its weight in fossil fuel.

In this study, we attempt to quantify the environmental impact of datacenters of various sizes, in terms of cost, materials and energy. We conduct our analysis under scaling over various datacenter sizes, S(mall), M(edium), and L(arge), in essence evaluating the effect of the economy of scale provided by Cloud Computing. We construct a realistic business scenario which we deploy over a typical Life-Cycle (LC) of datacenters. Our results show that the efficiency observed in larger installations yields a lower footprint than the footprint of smaller installations. In the following sections, we present a short overview of the deployment of the power and cooling equipment in datacenters, as well as our assumptions. We then calculate the costs and amount of materials for each deployment and conduct a comparative analysis among all deployments.

III. Background

In general, a typical cooling installation consists of one or more CRAC/H (Computer Room Air-Conditioner/Handler) units, a condenser, a chiller, and a heat exchanger. The purpose of the CRAC/H units is to provide the IT machinery with appropriately cold air, while removing the heat emitted from the machines. The condenser, in turn, serves for condensing the air into water, which the chiller cools down to a particular temperature. The chiller is connected to a heat exhanger, whose purpose is to exchange the heat produced by the chiller with some medium (water, air, or a coolant).

The choice of a particular solution depends on various factors (e.g. ambient conditions or architectural restrictions), but in this analysis we ignore these factors and simply consider the most efficient type of solution, which happens to be the water based one. Also, we make an assumption based on the fact that a relatively small server configuration (1-20 racks) is mostly likely to be located within a large building with an existing chilled water loop. Therefore, for all datacenter sizes $< S$ we assume that the CRAC units hook directly to a pre-existing water loop.

IV. Analysis

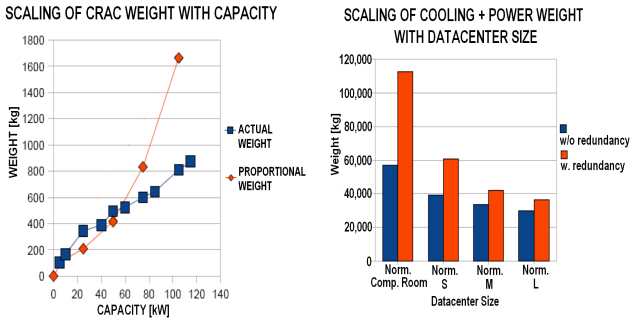
In this section, we analyze the capital (CAPEX) and the operational (OPEX) costs for each datacenter size, and compare the amount of materials used in the manufacturing of the respective cooling and power equipment. We find that both types of costs, in general, decrease with the datacenter size, except for the case where we compare the CAPEX of the individual computer rooms with that of the small datacenter (and find that it is 10.57% smaller). The scaling of the amount of materials, as expected, follows a similar trend with the CAPEX cost. The largest savings occur when we migrate from the small-sized datacenter to the large-sized one: the CAPEX drops by 23.72%, while the reduction in the amount of materials is 47.84%, without accounting redundancy.

A. Analysis of materials demand

In order to approximate the amount of the materials used in the manufacturing of the units, we use the weight that is

provided in the product specs. We plot the scaling of the weight with the capacity for the CRAC units. We want to compare this graph with a graph that depicts how in theory the weight would proportionately scale with the capacity. For our proportional weight graph, we only double the weight when the volume of the model changes. The actual rate of growth is indeed much slower than the theoretical proportional rate.

We next quantify the savings in the amount of materials for the cooling equipment (Chiller+CRAC) and power (UPS) among the various sizes of datacenters. The only case where a datacenter size requires more materials than the equivalent number of individual computer rooms is when we compare the latter with a small datacenter. This is because we assume that the individual computer rooms are hooked to an existing chiller of very large capacity, as it would be anticipated in the case of the deployment of a computer room in a large commercial building with an installed air-conditioning system. The savings in materials from deploying the equipment in a large datacenter as opposed to an equivalent number of individual rooms is 47.84% when the redundancy is N, and 67.81% when the redundancy is N+1.



(a) Growth of CRAC weight with capacity (b) Materials demand with datacenter-size

Fig. 1. Growth of the materials demand for the Coolind and Power equipment

B. Capital and Operational Cost Analysis

In order to evaluate the capital and production costs, we use a formula adapted from [3]:

TCO = Cooling and Power CAPEX + Cooling and Power OPEX,

where CAPEX is the capital investment made upfront and depreciated over time, and OPEX is the monthly incurring costs such as the electricity cost. Our adaptation of the formula consisted of removing the Server components, as the server requirements do not change across the cases, and we compare the magnitude as opposed to taking their ratios.

We assume a life-cycle of the datacenter of 10 years. We assume exponential growth of each of the business and e-commerce applications, as of doubling of their requirements every 4 years. Since during the deployment of datacenters the cooling and power equipment are typically chosen so as to accommodate the future and not the current requirements, we assume that the administrators of a small computer room will only purchase new cooling and power equipment every 4 years, while the administrators of a large facility will be upgrading this infrastructure much more often, i.e. every 6 months. As an exception for the large facility, the chiller is a component that should be upgraded infrequently, and therefore, we assume that the chiller will get upgraded every 4 years. We acquired prices for the equipment using Google's Shopping service. We then depreciated this cost over 4 years and added an interest rate of 8% as of [3]. By the end of the datacenter's lifecycle (10 years), we accumulate the depreciated cost with the interest rate, which is the total cost over the datacenter's lifecycle.

In Table I, we summarize the CAPEX and OPEX costs for all sizes. These results are normalized over the large configuration, for direct comparison. For the CAPEX, we observe that, except for the computer room configuration, the CAPEX decreases with the size. The reason why the computer room configuration costs less than the S datacenter configuration is because of our assumption that the Computer Room hooks to a preexisting building chiller-loop. As for the OPEX, we see that the OPEX is inversely proportional with the size, with the Computer Room configuration being a lot larger than everyone else. This is due to the inefficiencies of the various units (PDU, UPS, CRAC, and chillers), which do not scale well and therefore cause much higher ineffeciencies for the smaller sizes.

Size	CAPEX [USD]	Energy [MWh]
Comp.Room	6,063,835	687,400
S	6,704,600	243,500
M	5,426,400	238,000
L	5,114,280	223,900

TABLE I
CAPEX AND OPEX FOR ALL SIZES

V. Conclusion

In this paper, we conducted a detailed comparison among datacenters of various sizes in terms of operational and capital cost, and over the typical Life-Cycle of the datecenter (10 years). We were particularly interested in comparing these large configurations with individual configurations which a single small company would deploy. Our comparison focused on quantifying the operational cost in terms of energy, and the capital cost in terms of cost and amount materials, and quantifying the economies of scale of Cloud Computing.

References

[1] C. P. et al., "Cost model for planning, development and operation of a data center," Jun. 2005.

[2] J. K. et al., "A simple model for determining the true total cost of ownership for data centers," 2006.

[3] L. Barroso and U. Hoelzle, *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Mark D. Hill, University of Wisconsin, Madison, 2009.

[4] E. Williams, "Energy intensity of computer manufacturing: Hybrid assessment combining process and economic inputoutput methods," *Environmental Science and Technology*, no. 3822, pp. 6166–6174, Oct. 2004.

A Framework for Sketch-Based Interface Development

Jeffrey Browne
Department of Computer Science
University of California, Santa Barbara
jbrowne@cs.ucsb.edu

I. Introduction

Whether drafting a new musical composition or diagramming the airflow of a room, creative tasks often draw designers to a whiteboard. Here, users can quickly visualize and test ideas without any commitment, which is important since often our initial attempts fail or must be modified in some way.

Sketch-based interfaces such as a digital whiteboard, pen digitizer, or tablet interface have the potential to greatly expand a typical user's experience by generating additional information in response to the user's drawings. For example, when drawing digital circuit diagrams on a whiteboard, if the board could simulate the logic represented in the drawing, then the designer could tell much more quickly whether her idea was a success or a failure.

Though pen-based input hardware has existed for decades, sketch-based applications have yet to become mainstream. Part of the reason for the lack of adoption is the high learning curve of implementing an interesting application that uses sketch recognition. Up to this point, programmers wanting to build high-level recognition logic have had to begin essentially from scratch. Building an application to take in and manipulate directed graphs, for example, would require a programmer to learn the intricacies of handwriting recognition for labels, shape classification for nodes, and multi-stroke input for arrows. Thus it is understandable that interesting and complex applications utilizing sketch have largely failed in the beginning stages.

In this paper, we discuss work that we have done to soften the learning curve for sketch-based application programmers by implementing a framework for sketch application development. By using this framework, developers can compose and build upon other previous work in order to create complex recognition applications. We then discuss some of the challenges that motivate further research in this area.

II. Framework Details

At a high level, the framework consists of a central board manager (*board*) and background recognition tasks (*apps*) that communicate via *annotations* on strokes. As shown in figure 1, as a user interacts with the system, it proceeds in the following steps: first, a user draws a stroke (set of ordered points) in the main board area, then the board notifies each app of the new stroke. Here, each app can choose to tag a set of strokes with an annotation. For example, if a stroke looks like an arrow,

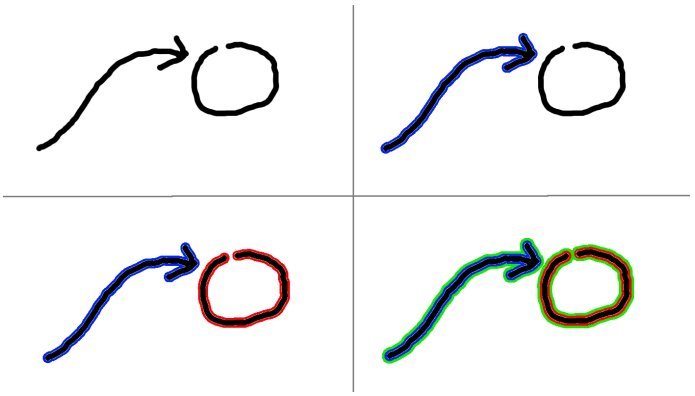


Fig. 1. Annotations building on two strokes. An arrow app annotates the first stroke as "Arrow" (blue) while the second stroke is annotated "Circle" (red). Then the graph app is alerted to an "Arrow" and a "Circle" and it annotates the two strokes together as "Graph" (green).

the "Arrow" annotation could have information about the head, tail, and length of the arrow. Similarly to listening for strokes, apps can also listen for annotations placed on strokes, and the board next notifies the correct apps of the new annotation. In the example above, after an arrow recognizer has tagged some set of strokes as "Arrow," the board would alert a graph recognizer app to the new annotation. Here again, the notified app may add annotations to some set of strokes, which will propagate similarly.

A. Erasure

When a user makes a mistake or wants to modify a design, he or she will often only erase some small set of strokes instead of clearing the entire board and starting from scratch. In order for our framework to support this, we must remove annotations and strokes from the board in such a way that the remaining information is semantically consistent with what is drawn. Currently, erasure in our system works on a stroke-level granularity, in that a user may remove entire strokes (e.g. an entire arrow) but not parts of strokes (just the arrow's head).

When a user erases a stroke, the system proceeds along similar lines to a stroke's addition. First, all apps that listen for raw stroke input are alerted that the stroke is disappearing. It is then up to each app to evaluate the annotations on that stroke and determine if they are valid or not; for example, a multi-stroke annotation for a large "Graph" should remain on the

board despite losing a single "Arrow" stroke. If it is no longer valid, the app will tell the board to remove the annotation. Then, any apps listening for that type of annotation will be alerted to its removal, where they will evaluate the validity of any annotations they may have added, and remove them as needed. Thus, annotation removal propagates in a similar way to addition.

B. App Design

Through our experience in designing and implementing several apps for this framework, we have found that some design patterns generalize to many applications, and we have implemented abstract classes to support them.

The first, and most basic function apps provide is as *markers*, which operate much like the arrow app discussed above. They simply analyze strokes and annotations, classifying them according to some rules, and then add annotations depending on the results. They do not remove any information from the board, and are a very basic form of recognizer.

Another style of recognizer that we have seen arise several times is a *collector*. These collect annotations into as few instances as possible, given some semantic rules. For example, a user might draw two separate graphs on the board, each of which would have its own "Graph" annotation. If the user then connects the two graphs via an edge, the graph recognizer should merge the two separate "Graph" annotations into a single instance, collecting the annotations into the single semantic object represented.

Factories are apps that wait for a certain set of rules, and upon matching them instantiates another app that will evaluate additional strokes on its own. An example of this is the "tic-tac-toe factory" which, upon recognizing the four-line hash of a tic-tac-toe board, instantiates a new tic-tac-toe recognizer that handles the state for that particular game.

Finally, as a way of organizing our system, we also currently distinguish *visualizer* apps. These apps observe annotations tagged to strokes, and then draw information on the board to communicate to the user. One strength of separating the visualizer functionality from the recognition apps is that multiple apps could potentially tag the same annotation type on different objects, but only one visualizer actually presents the information to the user. This cuts down on clutter and interface consistency issues.

III. Future Directions

Through the use of our framework, we anticipate developers will be able to tackle many of the more interesting problems in sketch-based interaction rather than reimplementing the same basic functionality repeatedly. In the future, we plan to evaluate the difficult areas of programming sketch-based interfaces more formally by studying how developers approach simple and complex problems in the area.

We also plan to study how users of sketch-based interfaces believe they should function. App designers will have to face many complicated decisions regarding user expectations and intuitive behavior, and our analysis should yield some practical suggestions. For example, erasure poses interesting problems with temporal ordering. If a user were playing a game of tic-tac-toe with a computer-controlled opponent, and all of a sudden erased his third "X," how should the system respond? The game could immediately be forfeit, or the user could be granted an extra turn, or the app could draw the strokes back in, effectively disallowing erasure.

A user could also erase a line making up the board itself. One option would be to garbage collect the state associated with the game, leaving other user strokes on the board (they may be part of some other annotations). Then if the user draws that line back in, the system could react in many ways, maybe by replaying the old moves, ignoring them altogether, etc. Our evaluation of what users actually expect in these sorts of situations will help us determine which solutions are most intuitive in the general case.

IV. Conclusion

The work presented here is a stepping stone to easing the development of sketch-based systems by allowing programmers to build on each others' code. Though this serves to organize existing code, app developers still have a long way to go in terms of accurate recognizers, intuitive feedback, and interesting applications. The effort will pay off in the end, because the unconstrained nature of user input that makes sketch interfaces so difficult to implement also makes them very powerful and expressive.

Active Cloud DB: A RESTful Software-as-a-Service for Language Agnostic Access to Distributed Datastores

Chris Bunch Jonathan Kupferman Chandra Krintz
Department of Computer Science, University of California, Santa Barbara
{cgb, jkupferman, ckrintz}@cs.ucsb.edu

***Abstract*—In this paper, we present Active Cloud DB, an open source Software-as-a-Service (SaaS) application that allows for RESTful access to cloud-based distributed datastore technologies that implement the Google Datastore API. We implement Active Cloud DB as a Google App Engine application that we employ to expose the Google App Engine Datastore API to developers – for use with any language and framework. We evaluate this SaaS on both Google App Engine and over AppScale, the open-source implementation of Google App Engine that enables Google App Engine applications to execute on cloud infrastructures without modification. As part of this work, we extend Active Cloud DB with simple caching support to improve the performance of datastore access and evaluate our technique with and without this support. We also make use of this support within multiple client-facing prototypes (e.g. Ruby on Rails, Python through Django) to show the ease-of-use and applicability of our contribution to other web development environments.**

I. Introduction

Distributed key-value datastores have become popular in recent years due to their simplicity, ability to scale within web applications and services usage models, and their ability to grow and shrink in response to demand. As a result of their success in non-trivial and highly visible cloud systems for web services, specifically BigTable [4] within Google, Dynamo [1] within Amazon, and Cassandra [3] within Facebook, a wide variety of open-source variations of distributed key-value stores have emerged and are gaining widespread use.

However, these datastores implement a wide variety of features that make them difficult for prospective users to compare. For example, there are differences in query languages, topology (master/slave vs peer-to-peer), consistency policies, and end-user library interfaces. As a result, we and others have investigated a single framework with which such systems can be compared. Others have done so through a system known as YCSB [6], while we do so through the AppScale cloud platform [5], [2].

AppScale is an open-source implementation of the Google App Engine cloud platform. It employs the Google Datastore API as a unifying API through which any datastore can be "plugged in". Once a datastore implementation is added to AppScale, it is deployed and configured automatically (there are command-line parameter settings for replication factor, cloud size, etc.) within the AppScale cloud deployment. Thus AppScale automates the configuration and deployment of these complex distributed systems and facilitates different databases to be compared. AppScale currently implements this key-value API using HBase, Hypertable, Cassandra, Voldemort, MongoDB, MemcacheDB, Scalaris, and MySQL Cluster. Unfortunately, AppScale only supports applications written in the languages that Google App Engine supports (currently Python and Java). YCSB does not support applications at all but instead spawns requests between the server and the datastore for the sole purpose of measuring datastore response time and throughput.

In this work, we address the problem of how to easily provide users of any programming language and framework with a database-agnostic interface to key-value datastores. To enable this, we design and implement a Sofware-as-a-Service (SaaS) component that runs over AppScale, called Active Cloud DB (in the spirit of Ruby's ActiveRecord). Active Cloud DB is implemented as a Google App Engine application that is implemented via the Google Datastore API and can run over AppScale and Google's environment.

II. Evaluation

We next employ Active Cloud DB over AppScale to evaluate the performance characteristics of the various supported datastores. We begin by describing our methodology and the present our results.

A. Methodology

For our experiments, we measure the performance of the primitive operations as part of an overall workload. In both scenarios, this is done over two back-end AppScale datastores, Cassandra version 0.5.0 and MemcacheDB version 1.2.1-Beta.

We use 16 nodes and 10000 random operations are performed with a 50/30/20 get/put/query ratio. Once an operation is selected, nine concurrent threads perform the operation and access their corresponding AppServers. We intentionally perform the operation on a single key in the datastore in order to maximize the amount of contention in the system.

As both of the datastores here have a number of settings that can be used, we configure both Cassandra and MemcacheDB in a particular way throughout AppScale deployments. While Cassandra allows the user to specify the consistency requirements on all operations, we set it to use inconsistent reads and writes: only one node in the system is needed to participate

in these operations. Conversely, in MemcacheDB, we direct all reads and writes to the master node in the system. While it does offer the ability to read from any database node, these initial tests access only the master node and use the replicas for data backup. Current work is underway to expand AppScale to read from any database node in the system.

B. Results

Figure 1 shows the performance of the system under a 50/30/20 get/put/query workload across 16 nodes. Get operations are faster for Cassandra than MemcacheDB, but now both are significantly slower than their cached equivalents. This is likely due to the substantially smaller amount of data in memcached, allowing for much faster read access. Write performance is roughly the same whether or not caching is employed.

III. Conclusion

We present Active Cloud DB, a Software-as-a-Service that runs over the Google App Engine cloud. Active Cloud DB is a Google App Engine application that executes over Google App Engine or over its open-source counterpart, AppScale. It exposes the Google Datastore API via REST to other languages and frameworks. We evaluate its use within Google and AppScale and present a number of proof of concept applications that make use of the interface to access a wide range of diverse key-value stores easily and automatically. We also extend Active Cloud DB with simple caching support to significantly improve query performance for Active Cloud DB and other Google App Engine applications that execute using an AppScale cloud. The proof of concept applications, AppScale, and Active Cloud DB, can all be found at `http://appscale.cs.ucsb.edu`.

References

[1] V. Bala, E. Duesterwald, and S. Banerjia. Dynamo: a transparent dynamic optimization system. *ACM SIGPLAN Notices*, 35(5):1–12, 2000.

[2] C. Bunch, N. Chohan, C. Krintz, J. Chohan, J. Kupferman, P. Lakhina, Y. Li, and Y. Nomura. An Evaluation of Distributed Datastores Using the AppScale Cloud Platform. In *IEEE International Conference on Cloud Computing*, 2010.

[3] Cassandra. http://incubator.apache.org/cassandra/.

[4] F. Chang, J. Dean, S. Ghemawat, W. Hsieh, D. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. Gruber. Bigtable: A Distributed Storage System for Structured Data. In *Symposium on Operating System Design and Implementation*, 2006.

[5] N. Chohan, C. Bunch, S. Pang, C. Krintz, N. Mostafa, S. Soman, and R. Wolski. AppScale: Scalable and Open AppEngine Application Development and Deployment. In *ICST International Conference on Cloud Computing*, Oct. 2009.

[6] B. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears. Benchmarking Cloud Serving Systems with YCSB, Mar. 2010. http://www.brianfrankcooper.net/pubs/ycsb.pdf.

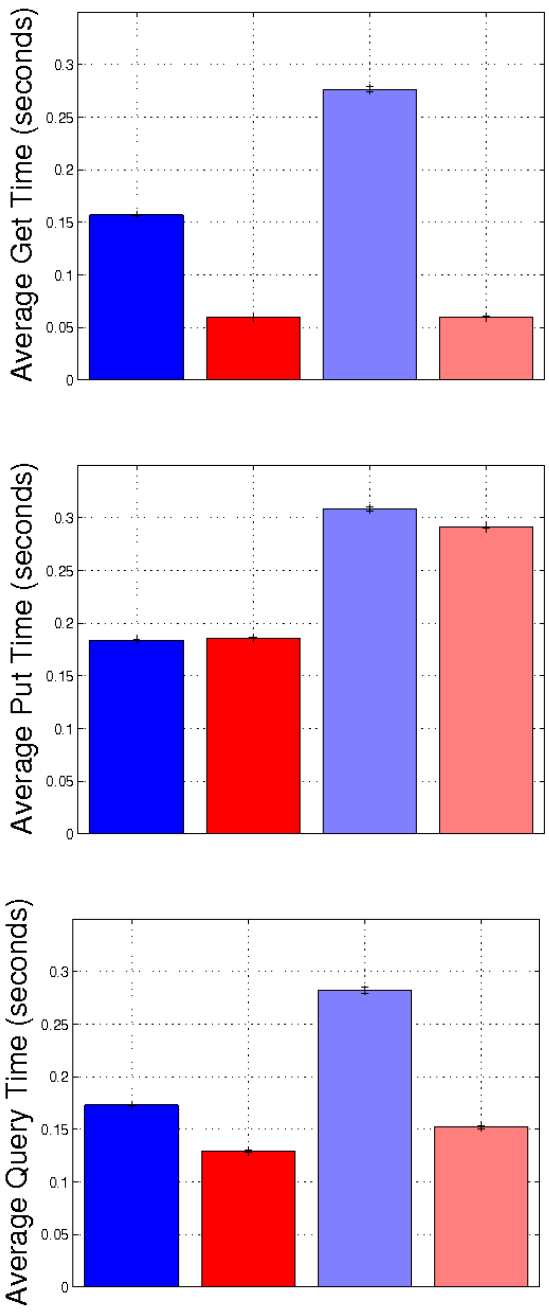


Fig. 1. Average round-trip time for get (top), put (middle), and query (bottom) operations under a load of 9 concurrent threads for a 50/30/20 get/put/query workload, run over 16 nodes.

Channel Management for 802.11n Wireless Deployments

Lara B. Deek, Kevin C. Almeroth, Elizabeth Belding
Department of Computer Science, University of California, Santa Barbara
{laradeek, almeroth, ebelding}@cs.ucsb.edu

I. Introduction

A considerable amount of work has looked at efficient methods of managing the available channels in wireless LAN deployments, in what is referred to as channel management. Channel management is the process of intelligently incorporating the characteristics of a wireless technology and its underlying environment conditions during channel assignment. Existing solutions, though exhaustive, have been designed to operate over traditional IEEE 802.11a/b/g wireless standards. With the advent of the IEEE 802.11n standard, a next generation wireless LAN technology, which adds major updgrades to legacy 802.11a/b/g clients [1], [2], new solutions must be designed to take advantage of the opportunities as well as deal with the challenges that this emerging WLAN technology provides.

The IEEE 802.11n standard has a number of new features that allow significant increases in data rate. These new features include multiple-input-multiple-output (MIMO) transmission schemes, channel bonding two 20MHz channels into a single 40MHz channel, as well as operating in the 5GHz frequency range which offers a larger number of subchannels to operate on in comparison to operating in the 2.4GHz range, as is the case for 802.11b/g [1]. Even in backwards-compatibility mode, where 802.11n does not utilize its additional features, 802.11n provides radio performance improvements with better communication range. With this knowledge and prior to even becoming Wi-Fi certified, vendors and users are accepting and quickly migrating towards 802.11n technologies.

Although 802.11n can potentially attain high data rates, given the number and complexity of its added features as well as the impact of the underlying environment conditions on performance, high data rates can only be achieved through intelligent and adaptive channel management strategies. For example, by accounting for the effects of MIMO and channel bonding together on channel performance, we are not only dealing with a channel assignment problem, where channels are assigned to particular users based on interference relationships, but also a management problem where all the characteristics of the technology are addressed together to achieve a combined level of performance. By addressing the impact of the (changing) wireless medium on system performance, the channel management solution is now also adaptive.

Channel assignment is an NP-hard, graph-coloring, problem that has been addressed in the context of 802.11a/b/g technologies [3], [4]. Existing channel assignment schemes attempt to allocate orthogonal channels to nodes that interfere with each other. A simple technique is to use static channel assignment. Recent work has addressed dynamic channel assignment solutions whereby changes in network characteristics, such as load or interference, trigger channel re-assignments [5]. Although multiple solutions to the channel assignment problem exist, they are limited by the fact that they do not exploit the flexibility and greater opportunities that 802.11n provides. There is a clear need to develop a comprehensive channel management strategy that can address the available features so as to maximize gains in channel capacity.

In this proposal, we discuss the characteristics of the IEEE 802.11n standard that need to be addressed in the design of a channel management solution. We also motivate the need for such a solution by showing how existing solutions in related research fall short in an 802.11n setting.

II. Background and Motivation

In this section, we first provide background information about the main 802.11n features, namely MIMO and channel bonding, and refer to related work that has addressed these features. We then discuss existing work on channel management solutions and load balancing. Through a discussion of background information and how related work falls short to 802.11n deployments, we motivate the need for a new channel management solution.

Multiple-Input-Multiple-Output (MIMO): Due to the availability of multiple discrete antennas, MIMO allows multiple data streams to be sent simultaneously along the same channel[1]. MIMO takes advantage of the multiplicity of data streams to improve data rate as well as at greater distances using *spatial beamforming* and *spatial multiplexing* techniques. Spatial beamforming coordinates and focuses the signal sent from each antenna on to a single receiver to maximize the signal strength at that receiver, hence increasing SNR to a single receiver at farther distances[2]. On the other hand, spatial multiplexing or diversity takes advantage of the multipath nature of wireless environments to send multiple spatial streams, hence increasing SNR. Althought the benefits of MIMO technologies have been well studied, the integration

[1]The draft-n specification provides for up to 4 spatial data streams; however, compliant hardware is not required to support that many.

[2]Broadcast and multicast messages do not benefit from spatial beamforming since they are not unidirecitonal in nature.

of MIMO into IEEE 802.11 wireless deployments have yet to be studied.

Channel Bonding: Channel bonding bonds two adjacent 20MHz channels together. 802.11n provides the option of operating over a 40MHz channel, which doubles the physical (PHY) layer data rate. The tradeoff of using channel bonding is that fewer channels will remain for other devices; furthermore, wider channels decrease range and are more susceptible to interference [6]. In traditional 2.4GHz Wi-Fi deployments where there are only 3 non-overlapping 20MHz channels, channel bonding was found to be more harmful due to limited channel availability and will lead to throughput degradation [7], [2]. On the other hand, due to the increased channel opportunities in the 5GHz range with 24 non-overlapping 20MHz channels and at least 12 non-overlapping 40MHz channels, there are more opportunities to exploit channel bonding in this frequency range. This flexibility in allocating bandwidth has defined the recent direction in bandwidth management solutions which advocates adapting channel width in wireless networks to accomodate changes in load conditions [8], [9], [10], [6]. New solutions should build over previous work and provide an analysis of the behavior of channel bonding under varying environment conditions, such as interference and traffic requirements. These solutions should also look at effective ways of incorporating channel bonding into channel management solutions in an adaptive and efficient manner.

Channel Management Solutions: Channel assignment in WLAN networks is a well-known NP-hard problem that has been sufficiently studied in legacy 802.11 technologies [11], [3], [12], [4] as well as in cellular networks [13]. Channel assignment attempts to allocate orthogonal channels to nodes that interfere with each other. Interference between APs or nodes in a network is modeled using a conflict graph (CG), where each node is represented by a vertex, and interference (or conflict) between two nodes is represented by an edge between the respective vertices. A CG is used during channel assignment to minimize the number of conflicting APs. As a result, the channel assignment problem is reduced to a graph-coloring problem, which is NP-hard [14]. Although multiple solutions to this NP-hard problem exist, they are limited by the fact that they do not exploit the flexibility and greater opportunities that 802.11n provides.

Load Balancing Techniques: The load within a network can vary significantly even on short timescales [8]. Therefore, related work has addressed improvements on the channel management problem by coupling channel assignment with load balancing. Proposed methods for distributing load over the network include manipulating client-AP association [15], as well as adjusting transmission power [4], [16]. Such approaches shift demand to lightly loaded APs, but weaken the RSS (received signal strength) and restrict throughput. In contrast, other work perform load balancing by shifting bandwidth to highly loaded APs using adaptive channel-width allocation [10]. Since 802.11n provides the option of operating over either a 20MHz or 40MHz channel, this inherent feature should be leveraged to perform load balancing in the channel management solution.

III. Conclusion

While efficient and low-overhead methods of building a channel management scheme for 802.11a/b/g networks have been well-studied in literature, there is a crucial limitation to the existing solutions if they are to be applied in 802.11n WLANs: existing solutions are not able to utilize and benefit from the enhanced channel options available in 802.11n. 802.11a/b/g follows SISO technologies and uses homogeneous channel widths of 20MHz. On the other hand, 802.11n can use MIMO technologies and channel bonding, and thus has greater opportunities to exploit the existing bandwidth. In order to design effective 802.11n channel management schemes, and given the complexity and variety of 802.11n communication parameters, proposed channel assignment schemes should take these characteristics into account. In this proposal, we discuss how we can leverage the vast accumulation of knowledge in the design of channel management solutions for SISO schemes and pick up from where previous work has left off: in the incorporation of 802.11n-specific features to best utilize the options for maximizing channel and network capacity.

References

[1] E. Perahia and R. Stacy, *Next Generation Wireless LANs: Throughput, Robustness, and Reliability in 802.11n*. Cambridge University Press, 2008.

[2] V. S. et. al., "802.11n under the microscope," in *ACM Conference on Internet Measurement (IMC)*, Vouliagmeni, Greece, 2008.

[3] N. Ahmed and S. Keshav, "SMARTA: A self-managing architecture for thin access points," in *ACM Proceedings of CoNEXT*, Lisboa, Portugal, December 2006.

[4] A. M. et. al., "A client-driven approach for channel management in wireless LANs," in *IEEE INFOCOM*, Barcelona, Spain,, April 2006.

[5] A. Raniwala and T. cker Chiueh, "Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network," in *INFOCOM*, Miami, FL, USA, March 2005.

[6] R. C. et. al., "A case for adapting channel width in wireless networks," *ACM SIGCOMM Computer Communications Review*, vol. 38, no. 4, pp. 135–146, October 2008.

[7] T. Instruments, "WLAN channel bonding: Causing greater problems than it solves," September 2003.

[8] R. Gummadi and H. Balakrishnan, "Wireless networks should spread spectrum based on demands," in *ACM Workshop on Hot Topics in Networks (Hotnets)*, Calgary, Canada, October 2008.

[9] H. R. et. al., "Frequency-aware rate adaptation and MAC protocols," in *ACM MobiCom*, Beijing, China, September 2009.

[10] T. M. et. al., "Load-aware spectrum distribution in wireless LANs," in *Internation Conference on Network Protocols (ICNP)*, Orlando, FL, USA, October 2008.

[11] X. Ling and K. L. Yeung, "Joint access point placement and channel assignment for 802.11 wireless LANs," *IEEE Transactions on Wireless Communications*, vol. 5, no. 10, pp. 2705–2711, October 2006.

[12] B. K. et. al., "Measurement-based self organization of interfering 802.11 wireless access networks," in *INFOCOM*, Anchorage, AK, USA, May 2007.

[13] I. Katzela and M. Naghshineh, "Channel assignment schemes for cellular mobile telecommunication systems: A comprehensive survey," *IEEE Personal Communications*, vol. 3, pp. 10–31, 1996.

[14] M. M. H. et. al., "On spectrum sharing games," in *ACM Symposium on Principles of Distributed Computing (PODC)*, Canada, July 2004.

[15] Y. B. et. al., "Fairness and load balancing in wireless LANs using association control," in *MobiCom*, Philadelphia, PA, USA, 2004.

[16] P. V. B. et. al., "Cell breathing in wireless LANs: Algorithms and evaluation," *IEEE Transactions on Mobile Computing*, vol. 6, no. 2, pp. 164–178, February 2007.

Identifying Communities with Coherent and Opposing Views

Nicholas Larusso, Petko Bogdanov and Ambuj Singh
Department of Computer Science, University of California, Santa Barbara
{nlarusso, petko, ambuj}@cs.ucsb.edu

***Abstract*—Collaboration capabilities are becoming pervasive in online applications. Social media and social knowledge systems are arguably samong the highest impact applications built around collaboration. Following its launch, Wikipedia has become the most visited not-for-profit free-content encyclopedia, attracting contributers and users who spend 5 minutes a day on average editing and accessing information. Little research has been directed towards understanding the multifaceted community structure of contributers and the topics they collectively promote.**

We propose a framework for discovery and characterization of collaborative community structure based on raw content-generation analysis. The key steps of our approach include (i) extracting signed pairwise contributor interactions (ii) modeling the editors community as a signed graph and (iii) analyzing the community structure in tandem with the content evolution. Our analysis reveals groupings of common-interest editors that back each other and collectively criticize other communities.

I. Introduction and related Work

Understanding how individuals interact and form communities in order to promote a common set of ideas has long been of interest to social scientists. The recent advances in social media (eg. Facebook, Flickr, Wikipedia, etc.) have provided researchers with a plethora of rich datasets that enable the large scale studies of social relationships, user interactions, group formation, and other phenomena of interest.

Our goal in this work is to identify user communities that share views on a subset of topics as well as those communities with opposing views. Being able to map such communities is important for ensuring content integrity and objectiveness. For example, if two groups with opposing views are actively editing an article, they can naturally regulate each other, thus producing unbiased content with little administrative overhead. In addition, using such a community model, new content can automatically be tagged as (non)controversial, based on the involved set of contributors.

Extreme polarity of opinions may also result in *edit wars*, manifested as sequences of adds and reverts of the same content. Such phenomena add overhead to the process of content generation as administrative resources must be allocated to arbitrate controversies. Prior understanding of opposing communities and their views may help automated conflict resolution, minimizing the overhead for administrators.

Much of the previous work in the area of detecting communities has focused on social networking data in which the network contains homogeneous and explicit links [3]. In these networks, a link between two users stands for a friendship relationship and often there is no differentiation between types of friendship (eg. high school friend, family relative etc.). In addition, networks comprised only of friendship links encode solely the positive part of the spectrum of individual relationships. Only recently have signed networks been used to represent and analyze the dynamics of communities, where links can be both positive and negative [2].

Previous work on community detection has been primarily directed to positive-relation social networks using traditional graph clustering techniques. However, in the presence of both positive and negative links traditional graph clustering approaches become inapplicable. A recent method [4] targeting signed graphs optimizes a clustering based on the positive edges and adjusts for the negative as a second step.

In this work we propose a method of constructing a signed interaction network purely from content, using text retrieval and topic modeling techniques. Additionally, we propose a method to discover communities in the context of the signed editor interaction network. Our approach, based on *Simulated Annealing*, naturally handles positive and negative interactions in tandem.

II. Methods

We would like to capture streams of opposing opinions within a single topic and identify the users who contribute to these streams in a text-based online collaborative system (eg. Wikipedia). We model the interaction between subsequent user revision at the level of different topical ideas within a single article. A natural separation within an article is the paragraph as separate ideas are developed in separate paragraphs. Note that the techniques we propose are not tied to this specific choice and similar analysis can be performed at the section or whole article level.

For this study we use articles from the Wikipedia edits dataset in which all revision of the articles are tracked. Each revision is comprised of a contributor, revision timestamp along with the state of the article after the contribution. For this study, we process the Anarchism article as it contains a large number of revisions and describes a controversial topic with an active group of contributors.

A. Scoring Editor Interactions

We propose an interaction model that is based on consecutive content modification. We represent any text excerpt using a bag of words model by applying punctuation and stop word

removal and term stemming, collectively referred to as *text retrieval*. Within this model, textual content is viewed as a high dimensional vector encoding term frequencies. Some terms are highly correlated while others have multiple meanings. To address these complexities between terms we transform the high-dimensional term representation into latent topic representation by employing a state-of-the-art probabilistic method called Latent Dirichlet Allocation (LDA) [1].

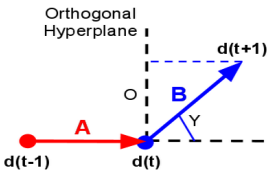


Fig. 1. Interaction of user B with user A

We determine the mode of interaction (positive or negative) based on the evolution of the content, undergoing consecutive revisions. This process is illustrated pictorially in Figure 1. The same text entity is moved from state $d(t-1)$ to state $d(t)$ by user A, and then to state $d(t+1)$ by user B. Each revision is represented by a corresponding revision vector which contains the changes in term frequencies from the previous state. In our example, B extends the topics that A previously extended along with some "orthogonal" topics and hence the angle Y between the revision vectors is acute. To score interactions, we use the cosine similarity score:

$$s(B,A) = \cos(B,A) = \frac{B \cdot A}{||B||\ ||A||}. \quad (1)$$

B. Community Identification

We combine user interaction edges into a global interaction network $N(U,S)$,where U is the set of all network nodes representing contributors and S is the set of all weighted interactions. When multiple interactions between the same pair of users are available, we represent them as the average of their scores. Note, that a model of the score distribution between two users may provide a more robust analysis (over sample mean) of users with multiple interactions. However, due to space, this model is not explored here.

Because our interaction network is signed and weighted, we would like our community identification algorithm to optimize two criteria: maximize in-community positive edges and cross-community negative edges. Formally, we define the energy of a clustering assignment $E(C)$ as:

$$E(C) = -\alpha \sum_i \sum_{A,B \in L_i} s(A,B) + (1-\alpha) \sum_i \sum_{A \in L_i, B \notin L_i} s(A,B), \quad (2)$$

where $s(A,B)$ is the weight of the link connecting nodes A and B, L_i is the node labeling for the i^{th} cluster, and α is a weighting parameter controlling the importance of positive and negative edges. In the extreme cases ($\alpha = 1$ and $\alpha = 0$), we are left with the standard formulation of a graph clustering criteria in which the goal is to maximize within community edges and the *Max-cut* problem respectively. Both of these extreme formulations are NP-complete problems.

The optimal community assignment is the one of lowest energy and the problem of finding it is an instance of the class of combinatorial optimization problems. We employ a simulated annealing (SA) based search to obtain an approximate solution.

SA works by proposing local changes in the community assignment. It accepts a change based on how it affects the global energy of the network. Unlike greedy methods, SA avoids getting trapped in local minima by probabilistically accepting changes that may increase the system energy. The rate at which 'bad' moves are made is controlled by the annealing schedule. This allows SA to effectively combine state space *exploration* and *exploitation*. When the temperature is high, a large number of states will be explored and as the temperature decreases, SA becomes more selective, only moving to states that reduce the system energy.

III. Experimental Evaluation

We apply our method on the contributers base of the Wikipedia Anarchism article. We use 25 topics for the LDA representation and produce a signed network of 577 users and 1896 interactions. Figure 2 present an evaluation of balance theory, postulating reciprocity of signed edges, for our experimental network. For clarity of presentation we quantize edges into strong negative (-2), negative (-1), positive (1) and strong positive (2). The figure shows the probability of having a reverse edge of certain type, given the existence of a forward one. Negative edges tend to have opposite negative and positive - opposite positive. This statistic alludes to the existence of collabotarion communities.

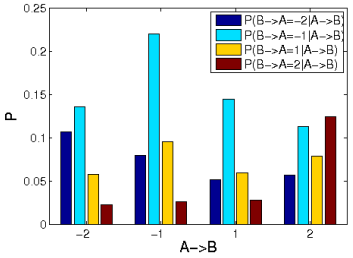


Fig. 2. Balance theory for Anarchism.

In addition, we have evaluated the convergence of our community discovery algorithm. We found that our algorithm consistently found local minima (with total energy values close to that of the optimal clustering). Results are not presented here due to lack of space.

References

[1] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3(4-5):993–1022, May 2003.

[2] J. Leskovec, D. Huttenlocher, and J. Kleinberg. Predicting positive and negative links in online social networks. *WWW*, pages 641–650, 2010.

[3] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. Aug 2003.

[4] B. Yang, W. Cheung, and J. Liu. Community Mining from Signed Social Networks. *TKDE*, 19(10):1333–1348, Oct. 2007.

A Study on VLSI On-line Stability Detectors

Chris Lee, John Oliver
Electrical Engineering Department
California Polytechnic State University
{clee83, jyoliver}@calpoly.edu

***Abstract*—Digital ICs become prone to stability faults caused by wear out effects like electromigration and hot electron injection over long term usage. These faults may cause timing violations to occur and eventually crash the system. Studies show that a typical chip is used for only a fraction of its expected lifetime before being discarded for this reason. Consequently, having a mechanism to measure these faults provides a way to scale down its operating frequency, rendering the chip reusable for another less computationally intensive application. Detecting these faults require concurrently running logic circuits called stability detectors. Several methods of on-line stability detection have previously been introduced, which involve probing the output flip-flop for erroneous signal switching between clock transitions. The main intention of this study is to characterize the performance of such stability detector circuits under various conditions including operating frequency, voltage, temperature and transistor sizing. The present major challenge in this study is to implement a configurable delay mechanism to provide test signals to the stability detector circuits. After analyzing several delay mechanisms, the most promising design involves a staggered parallel RC network formed by CMOS devices. Variations of this method are used in clock deskewing and correlating delay variations in high speed circuits. This paper presents the progress achieved up to date as well as the work scheduled for the following stages of this study.**

I. Introduction

Digital VLSI circuits operating at high clock frequencies are prone to stability faults over long term usage. Effects like electromigration and hot electron injection cause timing violations that would eventually crash the system. Previous studies [1] have highlighted that a typical chip is used for only a fraction of its expected lifetime before being discarded for failing to operate as initially intended. Consequently, having a mechanism to measure these faults provides a way to scale down the operating frequency of a chip, rendering it reusable for less computationally intensive applications.

Timing violations are typically caused by late input signal arrivals. Wear out effects cause MOS interconnect wear out, causing increased resistances that lead to longer RC delays. A study done in [2] describes the causes of wear out faults and outlines some useful wear out fault models.

In this paper, we report the work done on designs for both Franco and Yada's stability detector circuits as well as the findings from the research on configurable delay circuits. This paper is structured as follows. In Section II, we outline the architecture and functionality of the stability detectors that are being studied. In Section III, we describe findings on existing circuit designs that create controllable delay signals. Conclusions and expectations of this study are presented in Section IV.

II. Stability Detectors

The concept of detecting delay faults was first introduced as stability detection [3], where the input and intermediate signals of an output flip-flop were compared after the active clock transition for a specific period of time. This period, also known as the checking period, is at least as long as the hold time for the corresponding flip-flop. An error occurs if the input signal changes within this checking period and the stability detector circuit generates an error signal.

A stability detector typically works by comparing two signal values within a checking period. This behavior is akin to an XOR operation. Another variation of stability detector circuits [4] performs this XOR comparison by using a redundant stage to replicate the initial signal value. This saved value is then XOR-ed against the current signal value during the checking period. Both these signals are fed into a sense amplifier acting as a comparator, which then generates the error signal.

The main objective of this study is to implement these stability detectors in hardware and characterize their performance. This involves designing built-in signal generator circuitry to simulate stability faults during operation. At present, both stability detector designs have been implemented in SPICE and physical layout, while work is ongoing in modeling a suitable on-chip signal generator.

A stability detector is typically a logic comparator that activates after an active clock transition for the duration of a set checking period. Practically, the detector can be visualized as a charge storing element that gets discharged very quickly as some trigger signal activates. That trigger would be the signal experiencing a stability fault during the checking period.

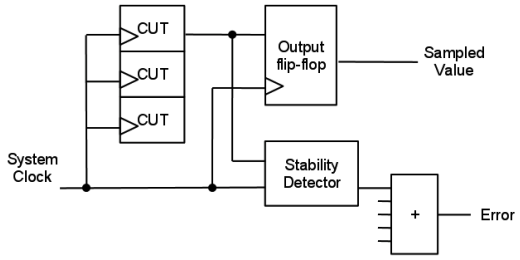


Fig. 1. Stability Checking Architecture

Figure 1 shows a single stability detector are located at the output stage of the functional unit to be tested. Each circuit-under-test (CUT) output requires its own stability checker which can be priority encoded in the next stage to determine the error source. Both system flip-flop and the stability checker cells share the common system clock. During the checking period for an active high clock, changes to the input signal will be detected by the checker, which then generates an error signal. Figure 2 shows the operation of the Franco stability detector [3]. In this simulation the late arrival of V(d) was detected during the second active clock pulse and the corresponding error signal was generated.

III. Configurable Delays

Simulating a stability fault requires generating a short delay that is at most as long as half a clock cycle. A controllable incremental delay generator would be essential for providing a comprehensive test pattern during characterization. In this section, three different delay mechanisms are briefly described.

Delay circuits are basically derivatives of RC networks. These circuits are constructed by exploiting intrinsic properties of MOS transistors which include resistive and capacitive regions within their structures. These RC structures contribute a transient delay whenever the transistor switches on or off, since each switching operation entails electrical charging or discharging.

At the gate level, additional transistors could be inserted at one of two logic gate inputs [5]. These transistors provide the RC delay component that creates a slower signal arrival time for one of the two inputs of the NAND gate. The length of the delay here is modified by manipulating transistor width(W) and length(L) values for the delay component: larger W increases capacitance while larger L increases resistance.

Other delay methods provide more flexibility in dynamic control. The techniques described in [6][7][8] create delays by enabling and disabling parallel pull up or pull down sections for an inverter buffer. These methods are logic restoring unlike [5]. Several models of these delay circuits were simulated in SPICE using TSMC 60nm MOSFET models.

Our preliminary study also included a cascaded logic gate delay chain delay technique. Each logic gate contributed a unit delay as the signal propagates through it. The outputs at each stage were multiplexed to enable a selectable line. The configurable delay lines will be controllable by a set of registers that interfaces with the user. A serial scan chain would be used to configure the delay circuit.

The results in Table I differentiate each design from one another particularly in terms of maximum delay range and average step size. Based on those values, we estimate the operating frequency of our system to be about 10GHz. The delay increments of the mechanisms would allow us to manipulate the Data and Clock inputs of the stability detector for the characterization process. In conjunction with signal delays, the characterization plans include varying the operating voltage and temperature of the system.

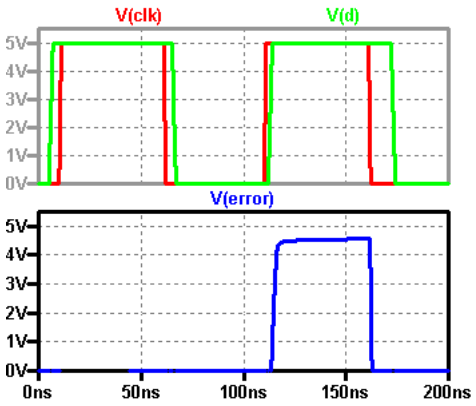


Fig. 2. Stability Checker Operation

TABLE I
Configurable Delay Comparison

Feature	Delay A[1]	Delay B[2]	Delay C[3]
FET count	38	27	90
Range	2.21ps	13.38ps	173.08ps
Step size	0.30ps	6.34ps	31.9ps

IV. Conclusion

The initial work done so far has provided most of the initial timing analysis data for the layout design phase which is currently underway. The following stages of this study entails selecting parameters for chip fabrication and packaging as well as board design.

References

[1] J. Y. Oliver, R. Amirtharajah, V. Akella, R. Geyer, and F. T. Chong, "Life cycle aware computing: Reusing silicon technology," *Computer*, vol. 40, no. 12, pp. 56–61, dec. 2007. [Online]. Available: http://works.bepress.com/jyoliver/4

[2] J. C. Smolens, B. T. Gold, J. C. Hoe, B. Falsafi, and K. Mai, "Detecting emerging wearout faults," in *In Proceedings of the IEEE Workshop on Silicon Errors in Logic - System Effects*, 2007.

[3] P. Franco and E. McCluskey, "On-line delay testing of digital circuits," *VLSI Test Symposium, 1994. Proceedings., 12th IEEE*, pp. 167 –173, apr. 1994.

[4] S. Yada, B. Amrutur, and R. A. Parekhji, "Modified stability checking for on-line error detection," *VLSI Design, 2007. Held jointly with 6th International Conference on Embedded Systems., 20th International Conference on*, pp. 787 –792, jan. 2007.

[5] R. S. Ethe, S. B. Commack NY, and S. NY, "Mos monostable multivibrator," US Patent US 4 629 908, 12 16, 1986. [Online]. Available: http://www.patentlens.net/patentlens/patent/US_4629908/en/

[6] S. J. C. Eddy C. Huang, "Cmos delay circuit with controllable delay," US Patent US 5 121 014, 06 09, 1992. [Online]. Available: http://www.patentlens.net/patentlens/patent/US_5121014/en/

[7] G. Geannopoulos and X. Dai, "An adaptive digital deskewing circuit for clock distribution networks," *Solid-State Circuits Conference, 1998. Digest of Technical Papers. 1998 IEEE International*, pp. 400 –401, feb. 1998.

[8] M. Maymandi-Nejad and M. Sachdev, "A monotonic digitally controlled delay element," *Solid-State Circuits, IEEE Journal of*, vol. 40, no. 11, pp. 2212 – 2219, nov. 2005.

[1] Delay circuit with controllable delay [6]
[2] Adjustable delay circuit [7]
[3] NAND gate delay chain

Secure Information Flow Analysis for Hardware Design: Using the Right Abstraction for the Job

Xun Li Mohit Tiwari Ben Hardekopf Timothy Sherwood Frederic T. Chong
Department of Computer Science
University of California, Santa Barbara
{xun, tiwari, benh, sherwood, chong}@cs.ucsb.edu

I. Introduction

Embedded systems are increasingly being used in critical applications that require a high level of assurance. For example, systems used in banks, automobiles, aircraft, and smartphones can benefit from strong guarantees on how secret or untrusted information flows through the system, ensuring that secrets never leak to unclassfied outputs or that untrusted information never affects critical system data.

To provide such guarantees, designers of embedded systems often rely on information-flow analysis tools. Information-flow analysis is a versatile technique that associates information labels (such as secret/unclassified or trusted/untrusted) with various system inputs, and tracks how these labels propagate through the system to the outputs. Such tracking can then be used to ensure policies on information-flow such as non-interference, which requires that secret inputs have no visible effect on unclassified outputs.

While there exist many techniques to track information flows through software [2], little work has been done to aid hardware developers in analyzing information flows through hardware designs. Ideally, hardware designers should be able to design using familiar idioms, get early design-time feedback about information flows in the system, and quickly iterate through different options to create verifiably secure hardware designs. However, hardware designs have unique characteristics that make a direct application of traditional information flow tracking techniques too conservative to be useful. Hence in this paper, we posit that *by carefully choosing the right level of abstraction for analysis, we can analyze information flows through hardware designs precisely using automated, language-level techniques.*

We explore a new direction where we base the information-flow analysis on a pervasive idiom in hardware design, namely Finite State Machines, and use this fact to make our analysis more precise. We propose that this trend towards expressing hardware explicitly as state machines not only retains the high-level programmability that is so desired by developers, but also allows more precise information-flow analysis.

II. Background

A. Hardware Design Using State Machines

Different from software design, in the hardware design field that state machines are widely recognized as a natural way to describe hardware controllers, and most commercial Computer-aided design (CAD) tools can extract state machines from Verilog/VHDL programs automatically. More recently, numerous state machine based languages and diagrams have been invented to explicitly express hardware as state machines [1], [3].

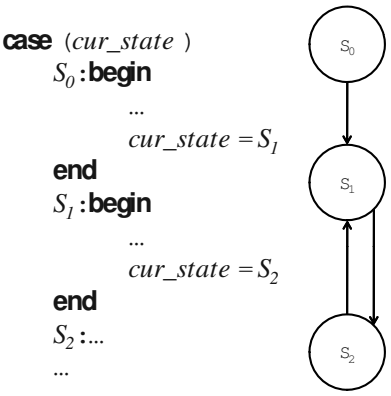


Fig. 1. **A Simple State machine Diagram.**

A state machine can be expressed as a set of states and transitions among those states triggered by *signals* which are either inputs to the state machine or local data. The most natural way to implement a state machine using programming languages is to have a variable `cur_state` to store the current state, and `case`-style statements to decide state transitions based on `cur_state` and some other conditions.

Figure 1 gives a simple state machine diagram along with corresponding Verilog program code. The state machine consists of 3 states S_0,S_1 and S_2, represented by the variable `cur_state`. Based on different value of `cur_state`, different behaviors are performed and then states are changed by assigning different values to `cur_state`.

B. Information Flow Analysis

When information flow analysis is applied to programs, all the variables are associated with security labels according to certain information flow policy. In a type-system based information flow analysis, those security labels are treated as types of the variables. Different typing rules are established to track different types of information flows: For an assignment statement $x = expression$, **Explicit Information** flows from

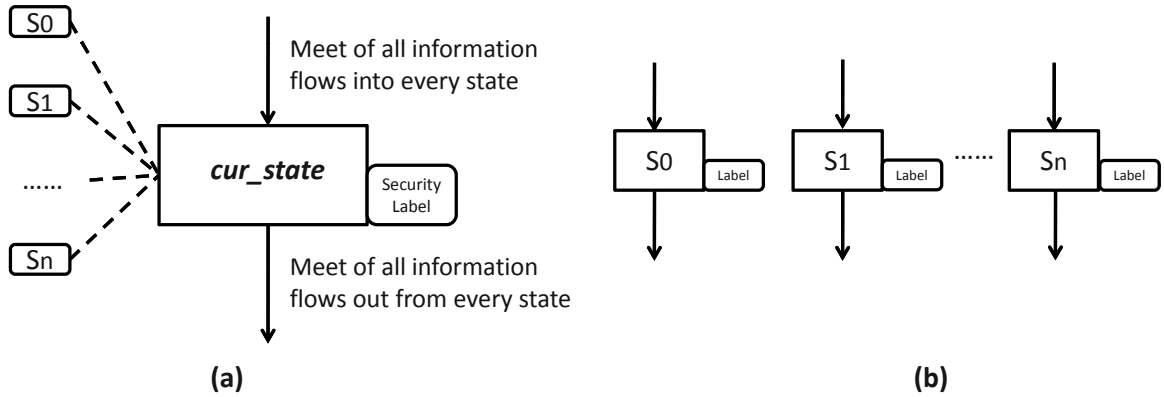


Fig. 2. **(a) Existing program analysis on state machines:** States are represented as values of a single variable `cur_state`, associated with a single tag, and all information flows into and out from every individual state are combined. **(b)Our proposed precise analysis:** Every specific state is analyzed independently.

the expression to the variable x. For a conditional statement $if\ (x)\ y = expression$, **Implicit Information** flows from the condition x to the assigned variable y inside the branch.

III. Hardware Description Analysis

A. Imprecise Program Analysis on Behavioral HDLs

Figure 2(a) shows how conventional information flow analysis is applied to state machine implementations. The value of `cur_state` can be one of S_0, $S_1 \ldots$, indicating the current state.

When information flows are analyzed, `cur_state` is associated with a single security label. Such analysis does not take into consideration the fact that information flows are actually flowing through each individual state, hence there is no way to track the security labels of individual states when states are represented only as different values of the variable `cur_state`. Whenever the variable `cur_state` get tainted/untrusted, everything will become tainted/untrusted according to the fact that there are implicit information flows from the conditional guard to the body. Such taint explosion makes the analysis conservative.

B. State Machine Based Analysis

The *key insight* of our approach is that by analyzing hardware descriptions explicitly as state machines (i.e., as a reified set of individual states with accompanying transitions) rather than as an implicit state machine encoded using variables, the analysis can precisely track security labels for individual states. Figure 2(b) shows that we associate security labels with each individual state, and analyze information flows for every state independently, hence we are able to derive more precise information flow relations.

C. General Framework of Our Tool

In conclusion, we propose to explicitly model hardware descriptions as state machines such that we are able to analyze information flows through every individual state, and give more precise results than conventional techniques. Figure 3 presents the general framework for our proposed approach.

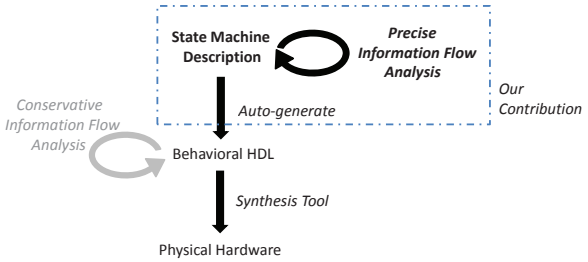


Fig. 3. **General Framework of Our Tool:** The highlighted part is our contribution which allows one to explicitly model hardware designs as state machines and perform more precise information flow analysis. The gray part– conventional imprecise information flow analysis techniques are then removed from the framework.

The bottom part represents the existing framework in which hardware descriptions are written at either behavioral or structural abstraction, verified by conventional analysis tools, then synthesized down to physical implementations. To enable precise information flow analysis, we add another level above behavioral hardware descriptions which allows one to describe hardware using state machine languages, verified using our proposed analysis tool and compiled to conventional behavioral or structural code using the tool's back-end.

The major benefits of such a static analysis based approach is that security properties are enforced and verified statically at design time, hence no runtime overhead will be introduced to the hardware. And type system based static analysis is known to be scalable as the size of the problem space increases.

Our future work seeks to build formal type system based analyses to express the reduction rules proposed in this paper, and explore potentials of such technique in software design.

References

[1] David Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming 8*, 1987.

[2] Andrei Sabelfeld and Andrew C. Myers. Language-based information-flow security. *IEEE Journal on Selected Areas in Communications*, 21(1), January 2003.

[3] Esterel Technologies. *The Esterel v7 Reference Manual, version v7.30 - initial IEEE standardization proposal edition*. 2005.

Overhead Reduction for a Gate Level Information Flow Tracking Processor

Van L. Nguyen, John Y. Oliver
Department of Electrical Engineering, California Polytechnic State University
{vlnguyen, jyoliver}@calpoly.edu

***Abstract*—With electronic security being a growing concern, cost-effective data tracking is becoming an important aspect of modern computer security. Recently, a Gate Level Information Flow Tracking (GLFIT) processors [1] has been found to be an effective way to track all data signals and determine which signals has been tainted and which can be trusted. General code injection, cross-site scripting, and exposure to a network are examples of attacks that can taint signals within the machine and affect its trustworthiness of sensitive data. As an extension of the GLIFT study, we are attempting to use both logical optimizations, as well as circuit techniques to minimize the hardware complexity of GLIFT.**

I. INTRODUCTION

Cryptographic keys, sensitive financial data, or even a list of client names are examples of sensitive information that potentially can be tainted by outside threats and foreign signals. A non-general purpose processor has be proven to be capable of tracking all information flow within a limited yet functional machine, including all explicit data transfer and all implicit flows. Though this processor has been proven to be effective in accurately tracking all signals and its state of trust (if the signal is tainted or not tainted), a drawback is its inefficient of hardware use.

A proof-of-concept processor was created using gate level information flow tracking. GLIFT introduces a unique processor architecture which utilizes a basic scheme that keeps track of a binary property (trusted or tainted) for every datum that flows through the machine. The main focus of this paper, with regards to the GLIFT processor, is that all information flows, whether implicit or explicit, are concrete logic function gates that replace weak ISA descriptions of general purpose processors but do so in a reduced overhead fashion.

In this paper, we present a few ways to reduce the hardware overhead of the GLIFT processor, yet keep the integrity of the proof-of-concept paper where this processor was conceived. With a large enough reduction in hardware, it may be feasible to manufacture the GLIFT processor and ensure computer security to a needing market. The follow will cover our means of measurement to compare each hardware reduction technique versus the original design, the actual gate reduction process and any conclusions and future works that can be made.

II. ALUTS

The Stratix II FPGA is the device used for the proof-of-concept and will also be used to evaluate the amount of hardware that can be reduced. Specific to ALTERA's Stratix family of FPGAs, hardware is measured in Adaptive Logic Modules (ALMs) which are divided into two Adaptive Look Up-Tables (ALUTs) [2]. The ALM contains a number of LUTs that can adaptively be divided between the two ALUTs; and since ALMs can have up to eight inputs for a combinational logic block, one ALM can do various combinations of two functions. Figure 1 (provided by Altera Corp.) depicts the relationship between a single ALM unit and its ALUTs as described above.

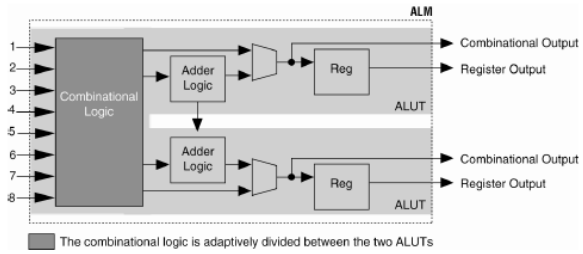


Figure 1. ALM is a logic array block that adaptively accommodates logic functions with different number of inputs and divides them between the two ALUTs

ALUTs contain combinational logic functions and registers. Focusing on the amount of combinational logic functions for each ALUT that is used is our measurement tool for area saved or lost. Since there is a direct correlation of logic devices to the physical number of transistors used in the FPGA, a lower amount of ALUTs used for the GLIFT synthesis will imply a lower amount of total transistors used, which physically means a smaller total area for the processor.

III. GATE REDUCTION

A. Proof-of-Concept Logic Design with Shadow Logic

The GLIFT processor is comprised of two parts; the processor which performs user's specified tasks, and shadow logic that tracks all the signals and its trustworthiness. Figure 2 shows an example of a simple OR gate and its Shadow Logic counter-part (which is consequently much larger in terms of sheer number of gates).

The main focus of this paper is to test techniques to reduce the number of gates or transistors used in shadow equivalents via logic optimization.

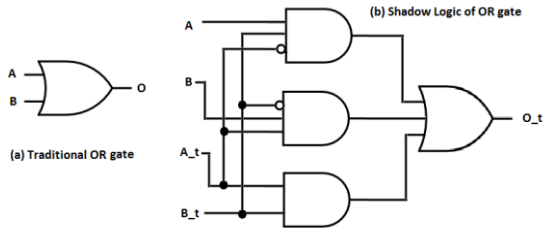


Figure 2. Side by side comparison of traditional OR gate (a) and the Shadow Logic OR gate(b) for precise data tracking and trustworthiness where A, B are inputs, O is the output, and "_t" indicates the trustworthiness state of the respective signal

B. Multiplexer Gate Reduction

Figure 3 presents the shadow logic from the proof-of-concept design for a general 2 input MUX. With knowledge that this processor uses MUXs as a part of its core design, with particular emphasis on the predicated architecture for handling conditional situations, the area of this shadowed MUX is quite large.

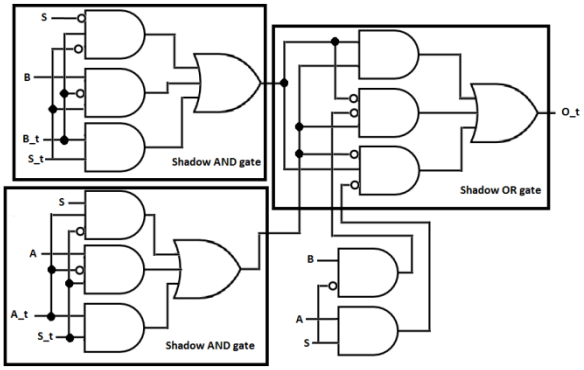


Figure 3. Original "Proof of Theory" Shadow Logic Equivalent to a GLIFT Processor's MUX which is made up two shadow AND gates and one shadow OR gate which mimics a standard MUX which has two AND gates and one OR gate

To understand the function of a combinational logic cell, we must map out its truth table. For the above cell, a six input truth table was evaluated and with any reduction that occurs, the logical output of the new cell must match the original to maintain functionality. With the shadowed MUX's truth table, an 8 X 8 Karnaugh Map was the preferred logic optimization technique used to yield the Boolean expression below.

$$(B + Sel + Sel\_t + B\_t)(A + Sel + A\_t)(\bar{A} + B + A\_t)$$

From this expression, a simpler logic cell can be made with fewer gates than the original shadowed MUX. Figure 4 shows the logic cell that maintains the original logical function of the shadowed MUX but with a reduction in the number of gates that are needed.

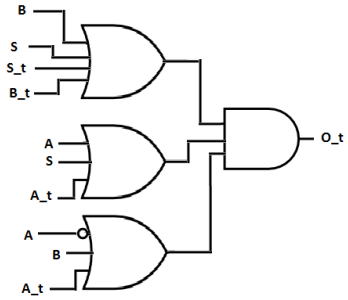


Figure 4. Reduced Shadow Logic for GLIFT Processor's MUX that was the result of logic optimization techniques and a direct representation of the Boolean expression found from the Karnaugh Map logic optimization technique

Another approach for hardware reduction would be a lower-level area-optimization technique. The above uses logic gate reduction techniques while maintaining its Boolean expression; but on a lower level, transistor arrays can keep the same logical output but have fewer over-all transistors. Complementary CMOS gate arrays are used as a circuit design technique to express Boolean equations without the use of logic gates. The idea of hardware reduction on the transistor level will produce a processor with a more reduced-overhead design.

IV. Conclusion

From the data that was gathered. There is a clear change in the number of ALUTs that were used in each design of gate reduction on the GLIFT processor. With the gate reduction process for the "Multiplexer Restructuring Statistics", there was a 378.2% decrease in the number of ALUTs used. Because of this large decrease in size, it may be viable to manufacture this processor on a custom chip. This approach of hardware reduction for the GLIFT design was proven possible and still maintained the original principle of complete traceability of all signals and its trustworthiness.

References

[1] M. Tiwari, H. M G Wassel, B. Mazloom, F. T Chong, and T. Sherood, "Complete Information Flow Tracking from the Gates Up," *Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems* (ASPLOS), 2009.

[2] Altera Corporation, "Stratix II Performance and Logic Efficiency Analysis," 2006.

Analyzing Ruby on Rails Data Models using Alloy

Jaideep Nijjar, Tevfik Bultan
University of California, Santa Barbara
{jaideepnijjar, bultan}@cs.ucsb.edu

***Abstract*—In this paper we demonstrate that data models in web applications implemented using the Ruby on Rails framework can be automatically analyzed using bounded verification techniques. In particular, we implemented an automatic translator from Ruby on Rails data models to the input language of Alloy, a SAT-based bounded verification tool. Alloy verifies assertions about object oriented data models by exhaustively exploring all possible configurations of the data model within a given scope. We experimented on two open source web applications to demonstrate the effectiveness of our approach.**

I. Introduction

Web application development frameworks based on Model-View-Controller (MVC) architecture have become the prominent style for web application development. By separating the data model from the navigation control and the view generation, these frameworks not only facilitate the use of sound software design principles such as modularity and separation of concerns, but they also provide an opportunity for static analysis and verification of the data model. In this paper, we focus on the Ruby on Rails (RoR) [2] framework, and we use an exhaustive bounded verification tool called Alloy Analyzer [1] for data model verification.

The data model specifications in RoR applications are based on Active Records [4], which define an object-relational mapping. We show that the data models specified as Active Records can be translated to Alloy [1], a specification language for object oriented models. After this translation, the developer can write assertions that the data model is expected to satisfy. We check these assertions within a given scope using the Alloy Analyzer. A scope simply bounds the number of objects in each class of the data model. Given a bounded verification problem, Alloy Analyzer translates each verification query to a Boolean satisfiability problem (SAT) and checks it using a SAT solver. If the property is violated Alloy Analyzer generates a counter-example, demonstrating an instance of the data model that violates the given assertion.

We wrote a translator that translates data models specified as Active Records to Alloy specifications. We conducted two case studies and analyzed data models of two open source web applications. Our experiments demonstrate that bounded verification of data models of real-world web applications is feasible.

II. RoR Data-Modeling Features

RoR uses an object-relational mapping based on Active Records [4]. Active Records handle all the details of connecting to the underlying database, mapping objects to tables, and data manipulation. Active Records are also used to manage relationships between tables.

In RoR, in order to create an object that is to be stored in the database (e.g. a `Person` object, with attributes `name`, `age` and `address`), one first defines a *database migration*, a change to the database schema expressed in a database-independent way. This creates a persons table with the columns `name`, `address`, and `age`). Then one writes the corresponding RoR model, which is automatically mapped to the database table created in the migration via its name. It is in these model files that the relationships between tables (objects) are expressed.

Active Record handles three basic types of relationships:

1) has_one: An ObjectA is associated with zero or one ObjectBs. (one-to-one relationship)
2) has_many: An ObjectA is associated with an arbitrary (zero or more) number of ObjectBs. (one-to-many)
3) has_and_belongs_to_many: An arbitrary number of ObjectAs are associated with an arbitrary number of ObjectBs. (many-to-many)

Each of these declarations have a set of options that can be set. The first is the :through option of the has_many declaration. The :through option is used when ObjectA has a one-to-many relation with ObjectB, ObjectC also has a one-to-many relation with ObjectB, and the RoR programmer would like direct access from ObjectA to ObjectC. So, rather than writing code to first get a set of ObjectBs from an ObjectA and then get the set of ObjectCs related to the set of ObjectBs, the programmer can directly obtain the set of ObjectCs from the ObjectA object.

The second option is the :conditions option, which can be set on any of the four declarations (has_one, has_many, belongs_to, and has_and_belongs_to_many). The :conditions option limits the relationship to those objects that meet a certain criteria. The condition statement needs to be in the form of the WHERE clause of a SQL query.

In order to express the inheritance relation in RoR, one uses the notation `ChildClass < ParentClass`. Typically all objects in the data model inherit from the ActiveRecord::Base class. This is so the data objects inherit all the database-connection functionality that is located in the ActiveRecord class.

III. Translation to Alloy

In R̄oR d̄ata model, the definition of the class looks like:

```
class MyClass < ActiveRecord::Base
... end
```

We convert this to the following statement in Alloy:

```
sig MyClass { ... }
```

where sig is the Alloy keyword for declaring a class. If the RoR class inherits from a class other than ActiveRecord::Base, such as:

```
class Child < Parent ... end
```

The corresponding Alloy statement would be:

```
sig Child extends Parent { ... }
```

Next, we discuss translating the three basic relationships in RoR: one-to-one, one-to-many, and many-to-many. When expressing a binary relationship in Alloy, one can provide a multiplicity of `one`, `lone`, `some`, or `set` which correspond to one, less than or equal to one, one or more, and zero or more, respectively. Thus, the mapping of the basic RoR relationships to Alloy is as follows:

class ObjectA < ActiveRecord::Base
has_one :objectB
end
	sig ObjectA {
	objectB: lone ObjectB
	}

class ObjectA < ActiveRecord::Base
has_many :objectBs
end
	sig ObjectA {
	objectBs: set ObjectB
	}

class ObjectA < ActiveRecord::Base
belongs_to :objectB
end
	sig ObjectA {
	objectB: one ObjectB
	}

class ObjectA < ActiveRecord::Base
has_and_belongs_to_many :objectBs
end
	sig ObjectA {
	objectBs: set ObjectB
	}

Furthermore, one has to add a fact block that connects each pair of declarations. (By default in Alloy, they are not related in any way.) For the one-to-many relationship this would look as follows:

```
fact { ObjectA <: objectBs =
~(ObjectB <: objectA) }
```

denoting that the two relations (the one from ObjectA to ObjectB and the one from ObjectB to ObjectA) are inverses of each other.

To translate the :through option of the has_many declaration, one follows the mapping as shown in the table, but instead of having a separate fact block one can simply add a fact block immediately following the signature of the object containing the has_many :through declaration. To translate the :conditions option, we create a subset of objects in Alloy which the object with the condition statement can map to. We omit the details here due to lack of space.

IV. Implementation and Experiments

To implement a RoR data model to Alloy translator we used a Ruby parser called ParseTree [3]. ParseTree extracts the parse tree for an entire Ruby class or a specific method and returns it as an s-expression. S-expressions are generated for each model file that contains a class that inherits from ActiveRecord. We then created an s-expression processor (which inherits from `SexpProcessor`, the basic s-expression traversal class provided with ParseTree) to traverse the generated s-expressions and translate them to a single Alloy specification file.

We used our translator for analyzing two open-source RoR applications, TRACKS [5] and Fat Free CRM [6]. TRACKS is an application to manage to-do lists. The to-do items can be organized by context or project; they can be starred and/or tagged; and notes can be added to them or a project. The to-do items can be given due dates, or they can be recurring items. Fat Free CRM aims to be a lightweight solution to customer relationship management (CRM). Fat Free CRM, offers the management of leads (a person who is a potential customer and usually represents an entire company), accounts (one created per customer) and opportunities; the conversion of leads into contacts (when a new customer is made); and the creation of campaigns which can include lead and opportunity generation.

We generated Alloy specifications from the data models of these applications and added assertions. We had two basic types of assertions. One was for checking the cardinalities of relationships, e.g. an objectA is related to one and only one objectB or that it is possible to have an objectA is related to no objectBs (in a zero-or-more relationship). The other type of assertion was regarding transitive relationships. For instance, if ObjectA is related to ObjectB and ObjectB is related to ObjectC, and there is also a relationship between ObjectA and ObjectC, then it is usually expected that the objectCs you get from an objectA object going through its relationship with ObjectB should be the same as the objectCs you get in the direct relationship between ObjectA and ObjectC.

All except three of the assertions we checked were verified by Alloy Analyzer. During verification we limited the scope to 3 and the longest verification time was 111 milliseconds per assertion. One of the failed assertions was in TRACKS where notes belong to users and notes also belong to a project. Projects belong to users. We asserted that the user a note belonged to was the same user that its project was associated with. This check failed, i.e. this constraint was not enforced in the data model.

V. Conclusions

In this paper we showed that RoR data models can be mapped to Alloy specifications. Based on this mapping we implemented an automatic translator from RoR to Alloy. Our experiments show that using our translator, assertions about RoR data models can be verified with Alloy Analyzer.

References

[1] Jackson, Daniel. "Software Abstractions: Logic, Language, and Analysis". Cambridge, Masachusetts: The MIT Press, 2006.

[2] Ruby, Sam and Dave Thomas. "Agile Web Development with Rails", Third Edition. Raleigh, North Carolina: The Pragmatic Bookshelf, 2009.

[3] "ParseTree". http://rubyforge.org/projects/parsetree/.

[4] Marshall, Kevin, Pytel, Chad, and Yurek, Jon. "Pro Active Record: Databases with Ruby on Rails". New York, New York: Springer, 2007.

[5] "TRACKS", http://getontracks.org/.

[6] "Fat Free CRM", http://www.fatfreecrm.com/.

Inferring File Structure from Disk I/O Traffic

Hunter Olson, John Oliver
Electrical Engineering
California Polytechnic State University, SLO
{hdolson, jyoliver}@calpoly.edu

***Abstract*—Utilizing the additional metadata and usage characteristics available to object based storage devices, aspects of hard drive performance can be optimized for the data contained on the device; however, implementing an object based storage system requires specialized hardware and software on the host machine and the storage device. We propose a method to identify objects and files on a hard disk from the record of block accesses, without any additional information provided by an OSD specific filesystem.**

I. Introduction

The Object Based Storage Device(OSD) moves many of the physical addressing and organizational operations normally located in the file system to the storage device itself. Using object IDs, a host can request flexible sized objects from the OSD without any concern over block addressing. Additionally, the OSD is able to store metadata for the objects, allowing the device to intelligently manage data based on usage statistics, Quality of Service concerns, or caching characteristics. In the case of a hard disk based OSD where performance is directly related to the physical location of the data on the disk, this additional metadata could be used to optimally place certain data on specific regions of the disk platters.

These OSD advantages are typically made possible by an OSD-aware file system on the host machine that sends requests by object-id rather than by block addresses. The T10 committee defined a subset of the SCSI standard in 2004 to standardize communication between an OSD-enabled host and an OSD. Although standardized, OSD is still a departure from the filesystem standards found in consumer machines today, so industry adoption of OSD in consumer devices is essentially non-existent. Because there can be no plug and play solution available without a significant push from the major companies controlling consumer operating systems, OSD must be implemented purely in the storage device itself for any benefits to come to market. To facilitate this, our research is focused on identifying file-like objects from only the standard block- level instructions provided to hard drives from the host.

II. Clustering Block Addresses

Given a list of instructions and the respective block addresses that they access(a trace), we seek to classify these block accesses into objects by identifying patterns in both their spatial and temporal locality. Looking at a graph of logical block address(LBA) versus issue time(Fig.1), we begin to see clear clusters of sequentially accessed blocks that very likely represent either entire files or pieces of files as seen by the file system.

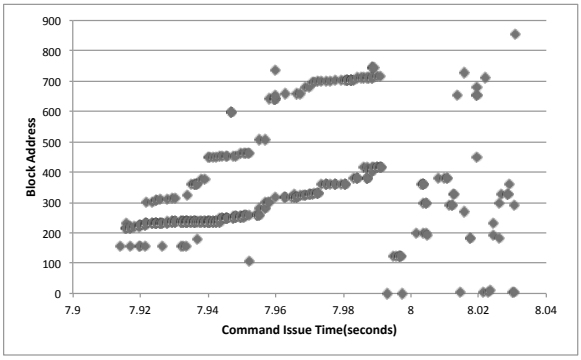


Fig. 1. Block Address distribution of 320 READ commands

There are many widely used clustering algorithms available, but memory and time constraints limit the effectiveness of many of the commonly used clustering algorithms when used with the extremely large data sets produced by a hard disk instruction trace.

K-means clustering is a popular clustering method that works efficiently with large data sets. However, K-means clustering requires that the number of clusters(the K-value) be specified at the start of the algorithm; this is not available in our case[1]. Techniques are available to predetermine the K-value, but most approaches optimize the k-value for modeling the distribution of the data. This approach did not yield K-values that correlate well with the actual number of detectable clusters that correspond to files.

Hierarchical clustering, in which clusters are created in a hierarchical fashion from previous clusters, is $O(N^2)$ in both memory and time, which is not efficient enough for the large data sets we are analyzing. The memory concerns exist because hierarchical clustering requires a distance matrix in which the distances between every data point and every other data point is contained. Our dataset, being time-series data, does not require this extensive distance analysis that results in quadratic memory usage. Our initial work has averted the complexity issue by reducing the dimensionality of our data.

III. Current Work

For our initial profiling efforts we focused on a recorded trace of the hard disk section of PCMark 05, a system benchmarking tool created by Futuremark Corporation. PCMark 05

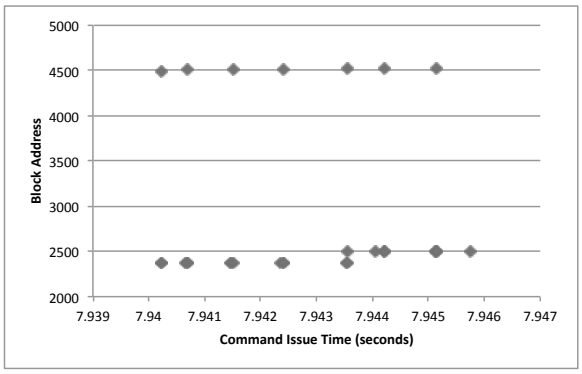


Fig. 2. READ commands from multiple threads

plays back a recorded trace of hard drive activity that is meant to replicate various types of everyday computer use and the associated disk access [2].

For an initial analysis, the time variable was removed and a simple one-dimensional clustering analysis of the accessed LBAs was performed on a section of the trace consisting of approximately 48,000 commands. This initial clustering effort looked for adjacent clusters of accessed LBAs that exceeded a minimum size threshold and were separated from other clusters by a minimum separation distance threshold. The minimum size threshold keeps each random access from appearing as a file, while the minimum separation distance prevents a series of files stored near each other on disk to be classified as a single object. By ignoring command issue time, this technique does not take into the account the effects of when the commands were issued in relation to each other, which, when considered, should greatly improve the correlation between the calculated clusters and actual files.

Because multiple threads can be accessing the storage device simultaneously, considering the time dimension alone yields inaccurate results. Fig.2 shows a series of read commands over 45 seconds that are sequential in time, but clearly represent two different files.

With knowledge of the file system layout, the storage device is able to make simple predictions about future commands. For example, once a request for a particular section of a file is received in a hard disk, the device can proceed to read the rest of the file into the buffer in anticipation of incoming sequential read commands. Our hard disk buffer simulations suggest that with basic pre-fetching of potential future blocks, there is well over a 6x increase in buffer hits. Some of this benefit is already realized in modern hard disks, but knowledge of the host file system should allow for benefit to be maximized.

IV. Future Work

In order to efficiently score the accuracy of our object identification algorithm, an instruction trace that includes both the block addresses and the corresponding files must be created. Using the logging capabilities of Linux and the included debugfs utility, such an annotated trace can be created. Once we have this data, we can tweak the parameters of the object classification/clustering algorithm until the results most closely match the known file structure corresponding to a given trace. Given a large enough set of training data, our clustering algorithm should be able to successfully identify most files that are being accessed by the host.

Because object identification is based heavily on the physical location of the block accesses on the disk, a heavily fragmented disk would significantly decrease the accuracy of the clustering algorithm. Our initial work assumes that the files are contiguous on the hard disk. As this assumption is known to be false in almost all real world scenarios, the effect of fragmentation on our algorithms performance will be evaluated.

Further work in cluster analysis is needed to improve the accuracy of classification of disk block ranges into their respective files. By splitting the large traces into smaller subsets, less efficient but more thorough cluster analysis can be used. Once clusters are determined from these subsets, the results can be compared and combined as needed.

V. Related Work

This work is inspired by much of the object based storage device research conducted in both industry and academia. Correctly implemented OSD filesystems have been shown to show performance benefits [3], which suggests that with the ability to classify incoming blocks requests into objects, storage devices could realize many of the same performance benefits with a non-OSD file system.

In order to correctly classify blocks into clusters that correspond to files on the host, an understanding of common read and write patterns is useful. Carnegie Mellon University has developed a statistical model to mimic the spatial and temporal correlation of real-world I/O [4].

VI. Conclusion

We are working to identify the file structure of a system given only the block level commands sent to the hard disk drive. This allows for many of the benefits of object-based storage devices to be embedded into the storage device without any additional complication added to the host filesystem.

Given the ability to identify files from a set of block level instructions, a storage device could optimize its performance by combining traditional file system optimizations with the low level control of the hard drive controller, resulting in higher performance than either mechanism working alone.

References

[1] Xu, Rui, and Wunch, Donald. (2005). Survey of clustering algorithms, IEEE Transactions on Neural Networks, 16 (3) 645-678.
[2] S. Niemela, “PCMark05 Whitepaper,” Futuremark Corporation, June 2005.
[3] F. Wang, S. A. Brandt, E. L. Miller, and D. D. E. Long, “Obfs: A file system for object-based storage devices,” Apr. 2004, pp. 283–300.
[4] M. Wang, A. Ailamaki, and C. Faloutsos, “Capturing the spatio-temporal behavior of real traffic data,” *Perform. Eval.*, vol. 49, no. 1-4, pp. 147–163, 2002.

A Study on Social Network Spam

Gianluca Stringhini, Christopher Kruegel, and Giovanni Vigna
Department of Computer Science, UC Santa Barbara
{gianluca, chris, vigna}cs.ucsb.edu

***Abstract*—Social networking has become a popular way for users to meet and interact online. Users spend a significant amount of time on popular social network platforms (such as Facebook, MySpace, or Twitter), storing and sharing a wealth of personal information. This information, as well as the possibility of contacting thousands of users, also attracts the interest of attackers. In particular, attackers might find personal information valuable for identity theft or to drive targeted spam campaigns. In this paper, we analyze to which extent spam has entered social networks. More precisely, we analyze how spammers who target social networking sites act. To collect the data about spamming activity, we created a large and diverse set of "honey-profiles" on three large social networking sites, and logged the kind of contacts and messages that they received. We then analyzed the collected data and identified anomalous behavior of users who contacted our profiles. Finally, we developed techniques to detect spammers in social networks, and aggregated their messages in large spam campaigns.**

I. Motivation

Over the last few years, social networking sites have become one of the main ways for users to keep track and communicate with their friends online. Sites such as Facebook, MySpace, and Twitter are consistently among the top 20 most-viewed web sites of the Internet, ranked just behind the most popular search engines. The many large social networks are even launching mobile platforms that allow users to access their services from mobile phones, making the access to these sites ubiquitous.

The tremendous increase in popularity of social networking sites allows them to collect a huge amount of personal information about the users, their friends, and their habits. Unfortunately, this wealth of information, as well as the ease with which one can reach many users, also raised the interest of malicious parties. In particular, spammers are always looking for ways to reach new victims with their unsolicited messages.

From a security point of view, social networks have unique characteristics. First, information access and interaction is based on trust. Users typically share a substantial amount of personal information with their friends. This information may be public or not. If it is not public, access to it is regulated by a network of trust. In this case, a user allows only her friends to view the information regarding herself. Unfortunately, social networking sites do not provide strong authentication mechanisms, and it is easy to impersonate a user and sneak into a person's network of trust. Moreover, it often happens that users, to gain popularity, accept any friendship request they receive, exposing their personal information to unknown people. In other cases, such as MySpace, the information displayed on a user's page is public. Therefore, anyone can access it, being friend or not. Networks of trust are important from a security point of view, because they are often the only mechanism that protects users from being contacted by unwanted entities.

Another important characteristic of social networks is the different level of user awareness with respect to threats. While most users have become aware of the common threats that affect the Internet, such as e-mail spam and phishing, they usually do not show an adequate understanding of the threats hidden in social networks. This behavior might be abused by spammers who want to advertise web sites, and might be particularly harmful to users if spam messages contain links to malicious pages.

II. Honey Profiles

In order to study the phenomenon of spam on social networks, we created 300 fake profiles on three popular social networking sites (Facebook, MySpace, and Twitter), and observed the kind of traffic they received. Due to their similarity in use to honeypots, we call these accounts *honey profiles*. After having created our honey-profiles, we ran scripts that periodically connected to those accounts and checked for activity. We decided that our accounts should act in a passive way. Therefore, we did not send any friend requests, but accepted all those that were received.

In a social network, the first action a malicious user would likely execute to get in touch with his victims is to send them a friend request. This might be done to attract the user to the spammer's profile to view the spam messages (on MySpace) or to invite her to accept the friendship and start seeing the spammer's messages in her own feed (on Facebook and Twitter).

After having acknowledged a request (i.e., accepted the friendship on Facebook and MySpace and started following the user on Twitter), we logged all the information needed to detect malicious activity. More precisely, we logged every email notification received from the social networks, as well as all the requests and messages seen on the honey-profiles.

III. Spammer Features

Network	Overall	Spammers
Facebook	470	73
MySpace	15	8
Twitter	341	320

TABLE I
Friend requests received on the various social networks.

Network	Overall	Spammers
Facebook	4,413	638
MySpace	20	0
Twitter	6,935	6,180

TABLE II
MESSAGES RECEIVED ON THE VARIOUS SOCIAL NETWORKS.

During our observation, we found out that spammer accounts differ from legitimate ones, and developed some features, we used later on for spam detection on these networks.

First of all, spammers usually send friend requests (or "start following", using the Twitter jargon) a large number of users, hoping that a fraction of these will follow them back, starting seeing the spam content on their walls. As a result, spam profiles usually have an unbalanced ratio between friends requests sent and actual friends. On Twitter these numbers are public, and the *following / followers* ratio constitutes a good feature for detection.

Another useful feature would be the ratio of URLs in the messages sent, compared to the overall number of messages. Since spammers have, by definition, to deliver some advertisement, it is likely that many of their messages will contain URLs.

From further observation of profiles contacting our honey accounts, two more features proved to be useful for spam detection. The first one is message similarity, which leverages the ideadetection. The rst one is message similarity, which leverages the idea that spammers send out messages similar in content. We also dened a *friend choice* feature as well, that attempts to detect whether a prole used a list of names to pick its friends or not. We dene this feature as the total number of distinct names appearing in the friend list, divided by the number of friends. Our observation showed that legitimate proles have values of this feature close to one, while spammers might reach values of 2 for it.

IV. SPAM DETECTION AND EVALUATION

We then leveraged our observation of spam behavior to build a system able to detect spammers on social networks. Given the described set of features, we built a system able to detect spammers in real time on Twitter. We tested our Twitter system by actively collaborating with Twitter itself. Whenever we detected a spammer, we submitted it to them to be checked. During a period of 3 months, from March 06, 2010 to June 06, 2010, we submitted 15,932 proles, and only 75 were detected by them as false positives. All the other submitted proles were deleted. In order to evaluate the false positive ratio, we randomly picked 100 proles, classied as legitimate by our system. We then manually checked at them, nding out that 6 were false negatives.

V. CAMPAIGNS

After having identied single spammers, we analyzed the data to identify larger-scale spam campaigns. With spam campaign we refer to multiple spam proles that act under the coordination of a single spammer. We consider two bots

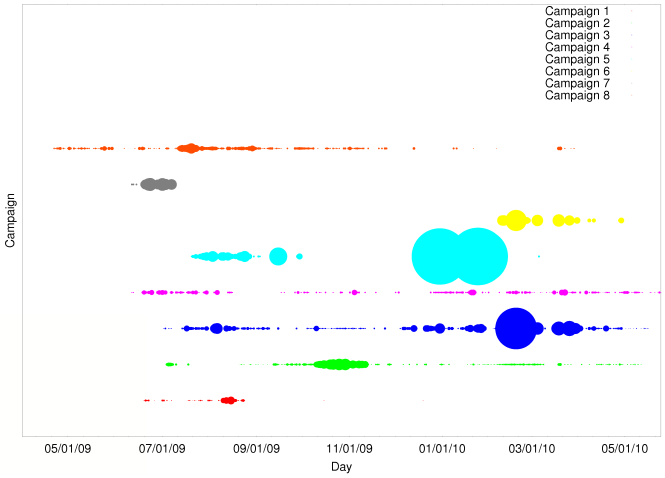


Fig. 1. Activity of spam campaigns over time.

posting messages with URLs pointing to the same site as being part of the same campaign. Some campaigns showed a large number of bots each sending a few messages per day, while others send many messages using few bots. In addition, some bots sent malicious content with each message, while others acted in a stealthy way, sending out a minority of spam messages, disguised among legitimate-looking messages. This behavior leads signicantly different outcomes. Greedy bots that send spam with each message are easier to detect by the social network administrators. On the other hand, a low trafc spam campaign is not easy to detect. Activity of bots from different campaigns is shown in Figure 1. Each row represents a campaign. For each day in which we observed some activity from that campaign, a circle is drawn. The size of circles varies according to the number of messages observed that day. As can be seen, some campaigns have been active over the entire period of the study, while some have not been so successful. The bot lifetime is affected by the campaign modus operandi as well. Some campaigns we observed had an average bot lifetime of more than 100 days, while some of them were easily detectable, and the bots belonging to them were deleted after a few days.

REFERENCES

[1] L. Bilge, T. Strufe, D. Balzarotti, and E. Kirda , All your contacts are belong to us. Automated Identity theft attacks on social networks, In *ACM WWW conference*, 2010.
[2] G. Brown, T. Howe, M. Ihbe, A. Prakash, and K. Borders,, Social Networks and Context-aware Spam. In *ACM Conference on Supportive Collaborative Work*, 2008.
[3] T.N. Jagatic, N.A. Johnson, and M. Jakobsson, Social Phishing. In *Comm. ACM, 50(10):94-100*.
[4] B. Krishnamurthy, P. Gill,and M. Aritt, A Few Chirps on Twitter. In *USENIX Workshop on Online Social Networks*, 2008.
[5] S. Moyer and N. Hamiel., Satan is on My Friend List: Attacking Social Networks, 2008.
[6] Harris Interactive Public Relations Research, A Study on Social Networks Scams, 2008.
[7] S. Webb, J. Caverlee, and C.Pu, Social Honeypots: Making Friends with a Spammer Near You. In *Conference on Email and Anti-Spam*, 2008.
[8] S. Yardi, D. Romero, G. Schoenebeck, and D. Boyd, Detecting Spam in a Twitter Network. In *First Monday, 15(1)*, 2010.

Characterizing the Potential of Chip-Scale Plasmonic Interconnects

Hassan M. G. Wassel Mohit Tiwari Luke Theogarajan* Fred T. Chong Tim Sherwood
Department of Computer Science *Department of Electrical and Computer Engineering
{hwassel, tiwari, chong, sherwood}@cs.ucsb.edu *ltheogar@ece.ucsb.edu

I. Introduction

In the multi and many core era, communication is crucial to the system performance. Therefore, network-on-chip (NOC) approaches were proposed to regularize the design of on chip communication. Nanophotonics interconnects have been proposed in the recent few years as a replacement of global metal interconnect because of their almost distance-independent power consumption and low-latency and high bandwidth. However, photonic components have their limitations: diffraction-limited sizes, temperature dependence and cross-talk and bend losses. First, nanophotonic components are governed by the diffraction limit which dictates that light cannot be confined in a space smaller than $\lambda/2n$ where λ is the freespace wavelength and n is the refractive index of the material. This means that the size of waveguides and modulators and ring filters are in the micrometer scale because of usage of C-band around 1550 nm wavelength. This size mismatch between micrometer-scale photonic components and nanometer-scale electronic ones limits the integration viability of both technologies on the same chip using the same process. This relatively bigger size components have higher capacitance which requires more power consumption to drive and limits the speed at which it can operate. For example, a photonic link is expected to have around 150 fJ/bit electric and electooptical components energy consumption per bit [1] which is clearly over the estimated viability requirement of 10 fJ/bit per device [4]. Second, photonic components is temperature-dependent which is exploited in the mirco-ring modulators that their resonance range is adjusted using heating. This heating requirement is estimated to consume around 100 fJ/bit [1]. These two major power consumption components limit the minimum distance at which nanophotonic waveguides can be more power efficient than electrical signaling, even excluding off-laser generator power consumption. Third, because of low confinement, bend losses and cross-talk are a problem for nanophotonic devices. The minimum pitch of a Si waveguide is 5.5 μm.

II. Surface Plasmon Polaritons Background

Surface plasmon polaritons (SPP) are electromagnetic waves that are coupled to free electron collective oscillations in a metal. When a light beam is incident a metal-dielectric interface with a certain angle, surface plasmon polaritons are excited and will propagate along the surface of the metal. Interestingly, surface plasmons excited at the interface of a metal and dielectric will maintain the frequency of the exciting light, while at the same time have a much shorter wavelength. These shorter wavelengths allow the construction of nanoscale waveguides that tightly confining even very high frequency electromagnetic waves, in a way side-stepping traditional diffraction limits. Of course nothing comes for free, and SPP propagation is limited by both metal absorption (i.e. ohmic losses) and free-space radiation. Noticing the importance of this emerging field, the term "plasmonics" was coined in 2000 . During the last decade, the plasmonics field has made significant progress in improving the the propagation distances of SPP modes and recently, we have seen a flurry of new work on both active components and plasmonic sources.

III. Plasmonics/Photonics Model

In this section, we evaluate the potential of plasmonics using an energy model that compares the energy-per-bit for the four technologies: electrical wires, photonics, fully plasmonic and hybrid photonic/plasmonic links.

A. Electrical Power Area Estimation

We used community standard Orion 2.0 to estimate the power consumption of electrical links. We used 32 nm at Vdd = 0.9 V high performance transistors. We limited the frequency by 3 GHz because that is the frequency at which routers power consumption is modest. Higher frequencies consume order of 1 W per router which is extremely prohibitive in a modern many-core interconnect. Orion reports power consumption that we divide over the clock rate to get the energy per bit power consumption. We assumed activity factor of 1 in all technologies.

B. Photonic Link Power Estimation

A photonic link consists of an off-chip laser source, transmitter, waveguide and a receiver. Power consumption is divided in static laser power and dynamic power consumption depending on transmitted bits. Laser power is estimated by adding all optical losses along the path and using the detector sensitivity (minimum detected power) to calculate the required laser power. Transmitters are ring resonator modulators driven by an analog circuit and receivers are a Ge photodetector connected to a TIA and an amplifier. Using parameters of analog components at 22 nm technology from [1], we estimate that analog components will consume 68 fJ/bit. Modulator is estimated to consume 82 fJ/bit at 32 Gbps as in [3] and 100 fJ/bit heating power as in [1]. For the receiver, we

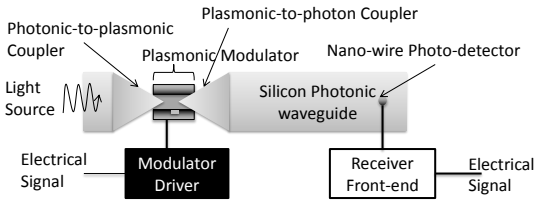


Fig. 1. Hybrid Link: By using a plasmonic modulator and silicon photonic waveguide, we can achieve the best of both world: long range propagation, and low power consumption and high performance modulation. Couplers converts photons into SPP and vice versa.

use the nano-wire photo-detector because it is not specific to plasmonics. That means that we assume 36 nW detector sensitivity.

C. Plasmonic Link Power Estimation

Because of the limited propagation distance of plasmonic waveguide to around 100 μm, we decided to connect more than one plasmonic link in order to achieve higher propagation distances. This means that we will have a modulator and a detector for each link, with detector of the current link driving the modulator of the next link. Analog components are modeled as in the photonic link and they are repeated for each sub-plasmonic link. Laser power is provided by a single photonic Si-waveguide for each sub-link and using a coupler between the plasmonic and photonic waveguides. We conservatively assume 4 dB coupler loss although lower coupling efficiencies has been demonstrated. We can calculate laser power consumption for each sub link. Dynamic power consumption is calculated assuming same elecrtic components used in the photonic link and the compact plasmonic modulator[2]. Number of sub-links is determined by the link length divided by the max propagation distance of a plasmonic waveguide. We assume plasmonic link loss of 0.2 dB/μm.

D. Hybrid Link Power Estimation

Finally, we propose exploiting active plasmonic modulation to modulate light in a photonic waveguide as suggested in [2] by coupling the the Si conventional waveguide to the modulator and use the modulated plasmon to couple it back into photons. This approach saves a lot of heating power required for photonic modulators that also consumes at least one order of magnitude higher energy per bit. It uses the same electric components and same photo-detector.

E. Characterization of Potential Applications

Using these models, we estimated the energy per bit of all four links by calculating the power consumption of all of them and dividing that by the corresponding bandwidth. We assume bandwidth of 100 Gbps for plasmonic and hybrid links, 32 Gbps for photonic links and 3 Gbps for electrical links. Operating photonic and plasmonic links at these high speeds assumes the availability of optical clock. Figure 2 shows the energy per bit consumption against the length of the link for all four configurations. Plasmonic link are modeled using 1 fJ/bit modulator driver and 10 fJ/bit receiver front-circuit and the parameters given in [1]. Plasmonic links cannot be more energy efficient than electrical wires at short

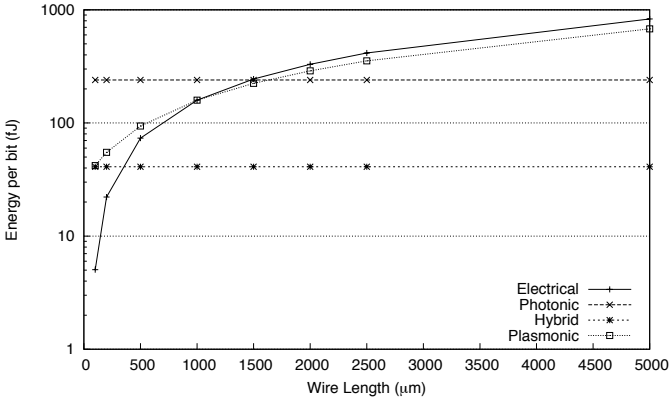


Fig. 2. Link length Vs Energy per bit for different technologies: electrical, photonic, plasmonic and hybrid. It is clear that electrical signaling is more efficient for any link of less than 500 μm length. Beyond that, hybrid links are the most energy efficient.

distances nor than photonic links at long distances because of two reasons. First, electrical links are really efficient at short distance where plasmonics don't suffer from the linearity of adding new electric components for each 100 μm. Second, plasmonics suffer badly from distance-dependent power consumption where photonics does not. However, between 1 mm and 1.5 mm, a plasmonic link shows comparable power consumption to electrical wires.

Using the plasmonic modulator with the conventional waveguide (hybrid) reduces the distance at which photonics become more energy efficient than electrical from 1.5 mm to less than 0.5 mm. This is an interesting result because we will use the best of both worlds, modulating using a small efficient plasmonic device and propagating the signal in a low-loss medium. It is worth noting that we lose the ability to support wavelength-division multiplexing because of the introduction of the plasmonic modulator that lies in the signal path whether it is modulating or not unlike ring-resonators that don't block the light if it is off-resonance. However, bandwidth loss is not big because of the ability of higher speed modulation (100 Gbps) and the space saved by the bulkier ring modulators. In order to study the effect of electric power consumption on hybrid links competitiveness, we did a sensitivity analysis in which we varied electric components power consumption from 12 to 100 fJ assuming 4 dB coupling losses. Results shows that for links of length less than 500 μm, electrical links are more energy efficient regardless of how aggressive the hybrid is. On the other hand, at 1000 μm hybrid is more efficient even with 100 fJ EPB.

REFERENCES

[1] C. Batten, A. Joshi, J. Orcutt, A. Khilo, B. Moss, C. W. Holzwarth, M. A. Popovic, H. Li, H. I. Smith, J. L. Hoyt, F. X. Kartner, R. J. Ram, V. Stojanovic, and K. Asanovic. Building many-core processor-to-dram networks with monolithic cmos silicon photonics. *IEEE Micro*, 29(4):8–21, 2009.

[2] W. Cai, J. White, and M. Brongersma. Compact, high-speed and power-efficient electrooptic plasmonic modulators. *Nano Letters*, 9(12):4403–4411, 2009.

[3] N. Kirman and J. F. Martínez. A power-efficient all-optical on-chip interconnect using wavelength-based oblivious routing. In *ASPLOS '10: Proceedings of the fifteenth edition of ASPLOS on Architectural support for programming languages and operating systems*, pages 15–28, New York, NY, USA, 2010. ACM.

[4] D. Miller. Device requirements for optical interconnects to silicon chips. *Proceedings of the IEEE*, 97(7):1166 –1185, July 2009.

Detection of Botnet C&C Communication Using Potential Signature Extraction

Ali Zand, Christopher Kruegel, Giovanni Vigna, and Xifeng Yan
Computer Science Department
University of California, Santa Barbara
{zand, chris, vigna, xyan}@cs.ucsb.edu

***Abstract*—Botnet, as a group of compromised machines under control of a single malicious entity, is a serious threat to online security. The fact that botnets by definition get their commands from a single entity could be leveraged to fight them by finding these entities. Detecting command and control messages is a complicated task, because botmasters sometimes use highly sophisticated methods to hide their command and control connections and make them look like legitimate traffic. In this paper a new pattern extraction method has been presented and used to find command and control communication in Anubis generated traffic.**

I. Introduction

A botnet is a collection of compromised machines controlled by the same command and control center. Nowadays, botnets are serious security threats to computer users and companies security. They are generally used to compromise other hosts, run DDoS attacks, perform fraud (like phishing, or click-fraud), steal private information (like bank accounts), or offer illegal services (like spamming).

The botmaster communicates (sends commands and receives information) with his/her bots using Command and Control (C&C) channel. Nowadays, P2P, IRC and HTTP are the most popular protocols for implementing botnet C&C channels [1].

One way to counter botnets is to detect their command and control communication and use this information to either filter the communication, to cut the commands from reaching the victims, to track the command and control servers and shut them down, track and prosecute the botmasters, take over the botnet, study the botnet (Stone-Gross et. al. [1]) or to disinfect the bots by issuing approperiate commands.

A signature extraction method, for detecting command and control communication in malicious network traffic, is presented in this paper. This type of analysis does not rely on synchronized activity of the bots, which makes it applicable to both real networks and also emulated malware traffic. To use this system we should have access to network traffic that has a significant amount of C&C traffic. This type of traffic could be obtained from honeypots, malware emulation tools (like Anubis), and also by filtering out known good traffic from a large network traffic data. This signature extraction method is most suitable for IRC and HTTP-based command and control channels.

Generalized suffix tree was used to extract all the frequent strings in connection payloads. Then, the connection clustering information was used to filter out non-distinguishing strings. A distinguishing string, is a string which appears frequently in one class of connections and infrequently in other ones. Non-distinguishing strings can not be good signatures, because they appear in a number of different classes of traffic, and they can not distinguish the application (for example a specific C&C protocol) generating the traffic.

Our hypothesis is that the connections from the same command and control communication protocol is recognizable from non-C&C applications and maybe other C&C protocols based on traffic characteristics. To cluster network connections, K-means algorithm was used on a set of statistical features extracted from the connections. Anubis network traffic was used as network traffic source. Anubis is an online dynamic malware analysis tool developed by the International Secure Systems Lab [2]. Anubis runs the submitted suspicious programs in an emulated environment and records the program interaction with the operating system and the generated network traffic. Currently, Anubis does not perform any type of analysis on the generated network traffic.

II. Connection Classification for Botnet Detection

We use a combination of frequent string extraction and connection clustering to extract distinguishing signatures for C&C messages. The whole process of signature extraction could be seen in Figure 1. In the subsequent subsections, different steps are described.

A. Common String Extraction

A Generalized Suffix Tree was used to extract all the common substrings with length more than 4 that occure more than 14 times in the traffic data. Many of the occuring patters are the ones that are indicative of the transport protocol which we are not interested in (strings like "HTTP", "GET", "POST", etc.). The connection clustering results were used to filter out the non-distinguishing patterns. The assumption is that different transport protocol patterns will exhibit different traffic characteristics, because they are carrying different applications data, and hence, they will apear in a lot of clusters.

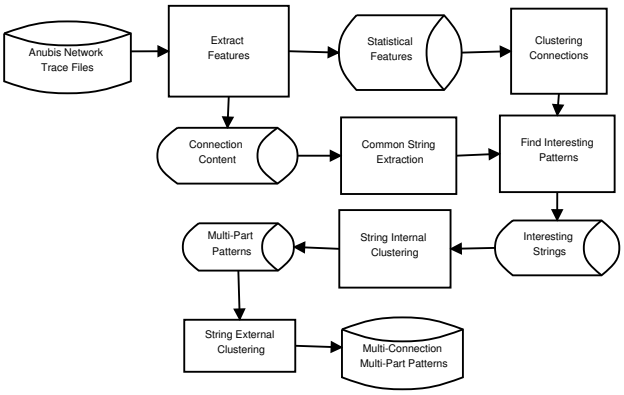


Fig. 1. Process Flowchart

B. Traffic Clustering

K-means clustering algorithm was used on the following statistical features of connections: connection duration, number of bytes sent by client/server (2 attributes), variance of data size sent by client/server (2 attributes), average/variance of inter-arrival time of the packets (2 attributes), number of packets with sizes [0-99, 100-199, 200-299, 300-399, 400-499, >=500] (6 attributes), whether the connection is initiated from/to remote site (two attributes), and the number of packets sent by client/server (two attributes). The Euclidean distance function was used as the distance function of clustering algorithm.

C. Non-Distinguishing String Removal

In this phase, the non-distinguishing strings were removed from the result set. Normalized entropy of the number of appearances of the string in different clusters was used to measure the quality of the patterns. The normalized entropy E_n could be computed using the following formula: $E_n = \frac{Ent}{Ent_{max}}$ where $Ent_{max} = Ent(unidist)$ where $Ent(unidist)$ is the entropy of the given string being distributed as uniformly as possible (all clusters having the same number of strings or at most 1 difference).

We used different values for normalized entropy threshold, and computed $precision$ and $recall$ of the pattern extraction algorithm and computed the optimum threshold value (maximizing $F - measure = 2 \times \frac{precision \times recall}{precision + recall}$ [3]).

We filter out the strings with normalized entropy greater than this threshold (in this case 0.7).

D. Combining Patterns

In this phase the strings that appear in the same set of connections are clustered together to represent a potential signature/pattern. This way, smaller signatures appearing in the same connections are combined to build larger signature strings which leads to less number of signatures with larger size, which makes the task of analyzer much easier to inspect these signatures. We used a modified version of Zamir et. al. [4] document clustering algorithm.

Pattern combination is done in three different ways: Combining patterns that appear in the same set of connections/malwares/destination addresses. The patterns that occur in the same set of connections can be a candidate for a signature for that type of connections. These patterns, when combined, form a multi-part pattern which is comprised on several strings that appear in different parts of the same connection. The patterns that appear in the same set of traffic files, are the patterns generated by the same type of malware, and can be a good candidate for a multi-connection signature. A multi-connection signature is comprised of several multi-part signatures from different connections. An example can be a signature for a bot, with some strings in HTTP connection and some strings in SMTP traffic. The patterns that appear in the connections with the same destination address, are good candidates for multi-version malware detection.

III. Conclusions and Future Work

A new signature extraction approach for botnet C&C messages was presented in this paper. Using this method we were able to detect known botnet C&C signatures in Anubis logs. The effectiveness of the approach will be significantly increased, if the running time of the samples increases and if each sample is executed several times. Unfortunately currently, because of lack of resources, increasing execution time, or several-time execution is out of question. This approach could be used only if the C&C messages are not encrypted.

References

[1] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna, "Your botnet is my botnet: analysis of a botnet takeover," in *CCS '09: Proceedings of the 16th ACM conference on Computer and communications security*. New York, NY, USA: ACM, 2009, pp. 635–647.

[2] U. Bayer, A. Moser, C. Kruegel, E. Kirda, U. B. (b, A. Moser, C. Kruegel, E. Kirda, C. Kruegel, and E. Kirda, "Dynamic analysis of malicious code," *Journal in Computer Virology*, vol. 2, pp. 67–77, 2006.

[3] C. J. van Rijsbergen, *Information Retrieval*. Butterworth, 1979.

[4] O. Zamir and O. Etzioni, "Web document clustering: a feasibility demonstration," in *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. New York, NY, USA: ACM, 1998, pp. 46–54.

Towards efficient medium access for 60GHz networks

Mariya Zheleva, Ashish Sharma, Sumit Singh, Elizabeth Belding, Upamanyu Madhow
University of California, Santa Barbara
{mariya, asharma, sumit, ebelding}@cs.ucsb.edu {madhow}@cs.ucsb.edu

I. Problem and Motivation

Millimeter-wave networks have emerged as an important research direction. This drive is a result of the advancement in the millimeter-wave circuit design, coupled with the growing need for short range high bandwidth wireless deployments. Networks operating millimeter-wave range are able to provide gigabit links in various wireless deployments by utilizing cheap *distributed* solutions.

The wave propagation in 60 GHz is fundamentally different from that in the 2-5 GHz range. First, these high carrier frequencies are significantly influenced by the Oxygen absorption, which results in *high path loss*. That is why we limit the radio coverage to a radius of 100m. Second, the transmission in the millimeter-wave range is *highly directional*, which introduces two important advantages: (i) it alleviates the high path loss caused by the Oxygen absorption and (ii) besides time and frequency, it fully utilizes a third degree of freedom: space.

Because of the directionality of 60 GHz antennas, the nodes are able to sense only in one direction at a time. This property causes the effect of deafness between neighbors. As a result, the traditional media access control approaches based on carrier sensing are infeasible for 60 GHz communication. Successful transmission demands that the transmitter and receiver antennas be both beam-formed towards each other at the time of transmission. Thus, precise node coordination is necessary for successful communication.

We develop a system that utilizes MDMAC [1] - a *decentralized* MAC protocol for 60 GHz networks. It provides a method for implicit node coordination, that allows convergence to TDM-like schedules for wireless networks in the millimeter-wave spectrum. We implement MDMAC framework on a commodity 802.11 hardware to evaluate its performance in a real testbed as well as to have the software support developed and ready for deployment once there is an actual 60 GHz platform available.

II. Background and Related Work

Previous work in the area covers interference analysis for 60 GHz networks [2]. The authors prove a level of link abstraction, that enables scheduling of transmissions between neighbors in the case of deafness - a characteristic that MDMAC design relies on.

III. Approach and Uniqueness

To the best of our knowledge, this is the first project that aims to investigate a MAC protocol suitable for highly directional 60 GHz networks via actual testbed study. We implement MDMAC [1] on a testbed of three x86 PCs running Linux and CLICK modular router [3]. The machines are equipped with Atheros 802.11a/b/g wireless cards and the MadWiFi driver [4], which allows higher wireless interface reconfigurability, needed for our implementation.

In the presence of immediate neighbor deafness, each node keeps track of the packet transmission history to achieve implicit coordination with the others. This history-awareness is referred to as memory-driven implicit node coordination.

Precise node coordination in a fully decentralized system demands accurate time synchronization. For this implementation we are utilizing *ntp* synchronization via Ethernet, which provides high enough level of accuracy.

In MDMAC, time is divided into frames, and frames into slots. For the purpose of implicit node coordination, each slot is assigned a transmission state, that defines the allowed sender/receiver interaction. There are four possible states - Transmit (T), Receive (R), Idle (I) and Blocked (B).

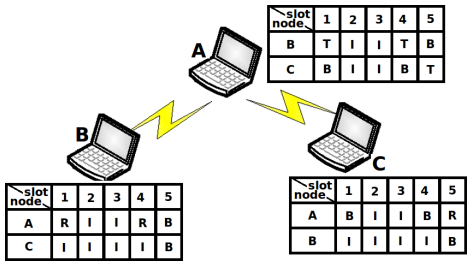


Fig. 1: Utilizing MDMAC.

A running example, illustrating implicit node coordination is presented in Figure 1. Node A attempts transmission to node B and it picks a free slot from its built-in transmission table. On successful reception of this packet, B sends an ACK back to A. The current slot is then marked as dedicated for transmission in both A and B. The process of packet/ACK exchange between A and B continues in this particular slot in the subsequent frames, until (i) there are no more packets left to be transmitted, (ii) B fails to send an ACK, or (iii) A fails to send a packet. After an initial set of iterations over the frame, all available slots are dedicated to the existing links

and eventually the network converges to TDM with no explicit coordination.

There are two main drawbacks of the outlined protocol, which limit the fairness and effectiveness of the communication. First, the lack of free slots for new nodes, causes what we refer to as unfair locked transmission schedule. Second, this solution does not address the problem for dynamic resource reallocation in case of variable link utilization, which might lead to inefficient medium access.

To prevent locked transmission schedules, nodes randomly reset their reserved and blocked slots. MDMAC also utilizes a mechanism for explicit state reset, which assures that in case of slot allocation overload, the most demanding links will be taken off slots first. Finally, for the needs of resource reallocation, each node keeps track of the utilization of its outgoing links and based on the information adjusts the number of utilized slots.

IV. Results and Contributions

In a TDMA settings the accuracy of slot transition is extremely important. Miscalculation of the current slot might result in severe packet losses and compromise the time-divided transmission scheme. To meet the high slot transition requirements we implement a *scheduler*, which makes decision for transmission towards specific neighbor based on the current slot.

A node running MDMAC supports multiple outgoing queues - one per neighbor. We have implemented these queues at the software level, so that the scheduler is able to pull packets from the corresponding queue once it determines the neighbor to transmit to. The multiple queues concept is also utilized by the explicit state reset and slot reallocation mechanisms.

To be able to maintain transmission states, all nodes in the network support data structures with information about the state of each slot towards each neighbor (Fig.1). As our slot state updates are based on packet and acknowledgements transmission history we implemented a software-level *acknowledgement scheme*. We also designed a *data structure* which maintains the slot-state allocation, as well as a *mapper*, which keeps track of the packet transmission and updates the data structure accordingly.

As the time scheduling is performed at the software layer, one of the major challenges that we tackle is the control over transition of packets from the software queues to the hardware. Hardware queues could store up to few hundred packets and the actual transmission is done at best effort, which might not be aligned with the upper layer schedule. To address this problem we estimated an upper bound of the number of packets that could be scheduled for transmission within a single slot. Our estimation is based on the expected time for transmission of one packet of certain size. In addition we have disabled random back off and retransmissions at hardware level to enhance the performance of our slotted transmission scheme.

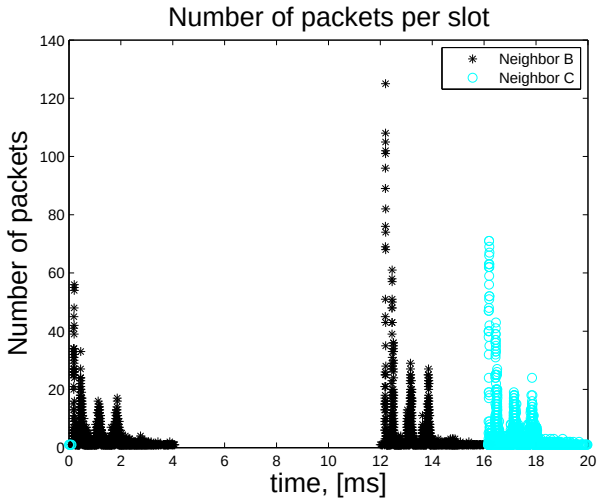


Fig. 2: MDMAC transmission accuracy.

In our experiments, the time frame is 20 ms and consists of 5 equal slots. Figure 1 depicts our actual test scenario, in which node A transmits to node B in slots *one* and *four* and to node C in slot *five*. Figure 2 shows the number of packets transmitted for one minute within each dedicated slot. As we perform best effort scheduling, the packets are pulled from the corresponding queue as soon as the proper slot comes. This explains the peaks in the beginning of each slot. The results show that transmission is accurately done as per the schedule. As expected, the number of packets sent towards B (assigned two slots) is almost twice as the number sent to C (assigned one slot).

Our contributions in this work are as follows: (i) we make an actual deployment to evaluate the performance of medium access protocol designed for directional 60 GHz networks, (ii) we implement the software support for a millimeter-wave network access scheme, so that it is ready for deployment once a millimeter-wave platform is developed, (iii) we provide a software-level control mechanism, not to allow hardware limitations to compromise our slotted transmission scheme.

References

[1] Singh S., Mudumbai R., Madhow U., *"Distributed coordination with deaf neighbors: efficient medium access for 60GHz mesh networks"*, IEEE INFOCOM 2010, San Diego, CA, Mar. 2010.

[2] Mudumbai R., Singh S., Madhow U. *"Medium access control for 60GHz outdoor mesh networks with high directional links"*, IEEE INFOCOM 2009, Mini Conference, Rio de Janeiro, Brazil, Apr. 2009.

[3] Morris R., Kohler E., Jannotti J., Kaashoek M. *"The Click Modular Router"*, ACM Transactions on Computer Systems (TOCS), Volume 18 , Issue 3, August 2000.

[4] *http://madwifi-project.org/*

http://gswc.cs.ucsb.edu