Aaron Schulman - Research Statement

Koomey's law—that the energy efficiency of computing systems doubles every 1.57 years—has allowed software development to be decoupled from hardware architectures for 50 years. However, as it becomes increasingly difficult to make marginal improvements in computing efficiency, **it is no longer sufficient to program for abstract machines.** We are reaching the limits of how far we can push existing hardware abstractions—and in some cases, existing hardware. I believe that a combined software and hardware approach is critical to resolving this problem. My research approach is to jointly develop hardware that is efficient and inexpensive as well as hardware abstractions that are intuitive for developers. I am uniquely suited to address this problem because of my expertise as a computer scientist and experience building embedded systems, circuits, and signal processing systems.

My work spans a wide variety of domains of computer systems, including networking, operating systems, and security. I have improved the efficiency of wireless networks, improved the flexibility of cellular networks, quantified the reliability of residential Internet networks, improved the energy efficiency of mobile applications, made measurable progress in securing the web's public key infrastructure, and identified mobile device privacy leaks.

A common thread in both my research and my teaching is bridging the traditional divide between hardware design and software development. This means I work at every layer of the stack, from the application down to the analog and digital circuits. I develop intuitive software abstractions of complex hardware and I also design and create simple embedded systems hardware to solve complex problems faced by protocol and software developers. By working in both of these domains, I am able to solve problems in one that make realistic assumptions about the other.

I also believe in engaging industry in order to inform my understanding of the practical limitations of hardware and software development. A concrete instance of this is the BattOr smartphone power monitor that I am in the process of commercializing, and is being used by Google and other companies. Also, Atomix, our software abstraction of multi-core Digital Signal Processing hardware, is being used by software developers at Deutsche Telekom to develop their own signal processing techniques for their next generation network.

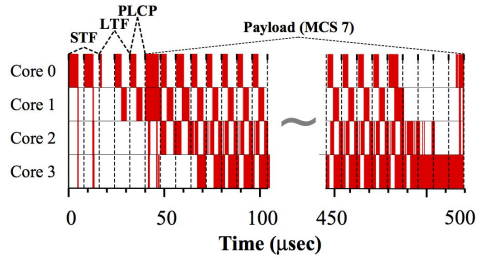
My research methodology can be summed up in three steps: (1) **measure** to find the root cause of complexity or inefficiency of software; (2) **identify opportunities** to improve the interaction between hardware and software to solve this problem; and (3) **design and evaluate** the new hardware and software to determine how well it solves the problem. I will now describe my existing research contributions that demonstrate how I apply this methodology, then I will describe my future work.

Research contributions

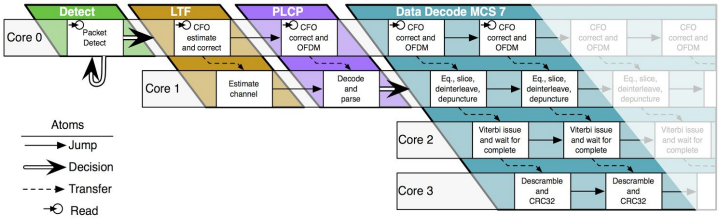
1. Mobile wireless networks are programmable, but they are difficult to modify.

Atomix [1]: Low power (5-8 Watts), heterogenous, multi-core Digital Signal Processor (DSP) System on Chips (SoCs) are the de-facto standard for radio signal generation and reception in today's cellular base stations. The wide deployment of these programmable processors presents an opportunity for deploying innovative signal processing applications on our existing cellular infrastructure. For example, we could deploy new signal processing techniques that are more efficient or add new features such as RF localization. However, deploying new apps requires a modular implementation of baseband processing, which is very difficult to implement due to the throughput and latency sensitivity of signal processing applications. My colleagues and I discovered that a simple software primitive, a block that has a fixed execution timing which we call an Atom, can be constructed in a modular fashion and thus provide latency guarantees without sacrificing throughput.

DSP core utilization while receiving a 1500 byte 802.11 frame transmitted at 54 Mbps at 10 MHz. Blocks have a variance in runtime of at most a half of a microsecond.



A detailed view of the execution pipeline of the Atomix 802.11 receiver. Notice how the predictable timing of processing each OFDM symbol of an 802.11 frame allows for a deep pipeline of the symbol processing across all cores.

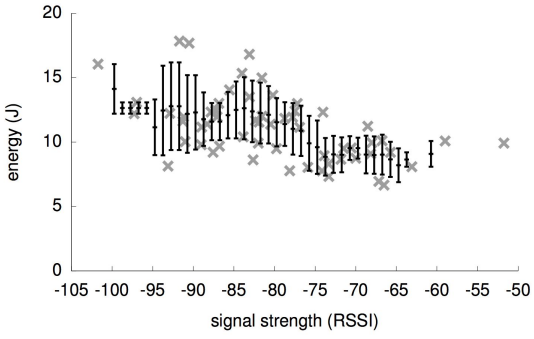


We implemented the 10 MHz 802.11 receiver signal processing with Atomix, and demonstrated that it is able to achieve 54 Mbit/s of throughput and a latency of 36.4 µs, with a variance of only 1.5 µs. We also demonstrated the modularity of the Atomix-built 802.11 receiver by modifying the cyclic prefix length to improve its robustness for long distance links and adding localization with angle of arrival calculation. We showed that the modifications had a predictable effect on the throughput and latency.

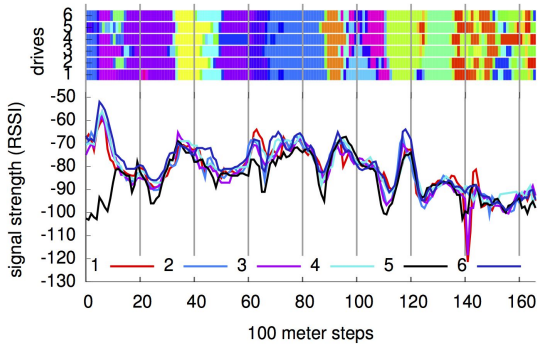
2. Abstractions of mobile wireless networking hardware are inefficient.

Bartendr [2]: The cellular radios in our smartphones are designed to operate efficiently, automatically adjusting the gain of the transmitter and receiver based on the needed signal strength to communicate with the base station. However, because software at the application level is not made aware of the current cellular signal conditions, inefficient communication happens even when traffic could be delayed until the user moves into a higher signal area (e.g., for an email sync or video buffering).

The average signal and energy for 60 SSL email syncs on a 3G EVDO device. Above -75 RSSI, the sync energy is consistently low. Below this threshold, the sync energy is higher and more variable.



When traveling along the same path several times, the cellular signal strength is stable in each of the locations. The colors of the top bars indicate the cell tower in use. They show that handoffs are consistent.



We measured the energy consumption of cellular modems and found that they can consume 6X more energy when operating in areas with low signal strength than when in areas of high signal strength. This means that if applications know the predicted signal strength for the next few minutes, they can avoid wasting energy by communicating when the signal strength is low. Finally, we designed algorithms for scheduling communication based on these signal strength predictions and evaluated their efficacy. We observed that this system saves up to 10% of the energy consumed by email sync applications and up to 60% for video streaming applications.

Maranello [3]: In an 802.11 network, when packets are lost due to interference or noise, the transmitter must send a full packet retransmission. Prior work tried to recover the correct portions of these packets. This is known as partial packet recovery. However, all prior work had significant drawbacks, including requiring the devices to disable their retransmission backoffs, making them incompatible with 802.11.

My colleagues and I studied the bit error patterns in these corrupted 802.11 packets and discovered that errors tend to appear in contiguous blocks in some portions of the packet. We then asked if it was possible for a receiver to calculate checksums for blocks of a packet within the 10 microseconds before transmitting the ACK. If the packet's CRC check failed, the receiver would transmit the checksums instead of an ACK.

We developed and evaluated a system called Maranello that demonstrates that this block-based recovery can be implemented in the firmware of existing off-the-shelf Broadcom wireless cards. Maranello improves throughput up to 2X by significantly reducing the number of retransmission attempts and backoffs that the transmitter must undertake to successfully transmit a packet.

4. Unicast is not well suited for distributing timely data to every device in the world.

RevCast [4]: The security of the Internet depends on users knowing not only whether they should trust an encrypted connection to a server, but also whether that trust has been revoked. Techniques for issuing trust have received a significant amount of attention over the past 20 years, but a fast and reliable method for disseminating revocations of trust has remained elusive. My colleagues and I performed several measurement studies of the current state of the trust revocation system on the Internet **[5, 6]**. We found trust authorities do not revoke trust when the should—even, for example, weeks after the massive Heartbleed vulnerability was announced.

My colleagues and I studied the current systems for distributing revocations and concluded that the main problem with these systems is not the technique they use to deliver the revocations, but the underlying transport they rely on. We found that unicast is not well suited for distributing revocations, because revocations need to be broadcasted—every device needs to know about a revocation as quickly as possible.

As a result, we investigated alternative transport that could provide these characteristics. Surprisingly, we found that an existing form of communication, metropolitan FM Broadcasting (specifically the 421.8 bits/sec available from the digital FM Radio Data System), is well suited to distributing broadcasts. Our system called RevCast consists of a protocol based on multi-signatures that enables broadcasting revocations over the low bandwidth FM RDS link. We implemented and evaluated a receiver circuit this system. We found that even meager bandwidth can handle the load of revocations on the Internet today, and even when there is a Internet-wide disaster like Heartbleed, it can disseminate every revocation in a matter of 15.5 minutes, compared to the hours or days of today's unicast based revocation systems.

Future Work

Developing a principled approach for improving the energy efficiency of mobile systems.

I believe the next frontier for significantly improving mobile devices is addressing their energy efficiency. Recent surveys[1,2] have found that battery life was the most common factor users consider when purchasing a smartphone, and developers have begun to respond. For instance, Google developers have been improving Chrome's power consumption by finding and patching energy bugs. . Also, ARM has resorted to a big.LITTLE CPU architecture aimed at improving energy efficiency by alternating between low-power cores for light processor workloads and high-power cores for heavy workloads.

In response to user demand for increased battery life, hardware and software developers now include energy efficiency as a first-order design goal. However, unfortunately, there is no principled approach to

[1] C. Arthur. Your smartphone's best app? battery life, say 89% of Britons. *The Guardian*, 2014.
[2] B. Reed. Battery life has become the single biggest reason people choose a smartphone. *BGR*, 2015.

developing an energy-efficient computer system. There are three facets to this problem: (1) on battery-powered mobile devices, developers lack the proper instrumentation of power consumption for regression testing, comparison with other implementations, and finding opportunities for improvements, (2) even with proper instrumentation, there is no structured approach to identifying the specific software module that is causing energy inefficiency, (3) even if they find the module that is inefficient, their existing development paradigms may be limiting the system's energy-efficiency. One proposed way to address some of these problems is to build fine-grained power monitors into every device, however, this introduces privacy and security risks. My research will addresses all of these issues.

Ideally, developers would have fine-grained power measurements from a wide variety of devices running in different environments. However, fine-grained measurements are not widely available. Instead, developers rely on models of the components' power consumption. While these models have proven useful in finding coarse-grained hotspots that can be made more energy efficient, they do not capture the complexity of a device's components (e.g., the energy required to render frames on the GPU). My insight is that a fine-grained power monitor circuit can be made small enough to carry with the device under test and small enough to deploy in a testbed of thousands of smartphones. I also have found that there are only a few types of battery connectors in our devices, so by creating a few battery interceptors we can get fine-grained power measurements from a wide range of devices. I have developed a power monitor instrument built on these insights called "BattOr". Using BattOr as their primary energy instrumentation tool, Google has already discovered and fixed a 30% power regression in Chrome on Mac's video playback.

I will start by building a mobile device power profiling testbed using BattOr power monitors. The impact of this testbed could be far-reaching; researchers and practitioners will be able to evaluate the energy efficiency of their applications on a wide variety of devices. The primary research challenge in creating this testbed is reproducing the conditions that users are likely to experience. I will investigate techniques for automatically generating representative and realistic energy benchmarks based on collected traces of user activity. I also will investigate the tooling (e.g., controlling thermal frequency regulation) that is necessary to produce reproducible runs of these benchmarks on many devices.

Even with the fine-grained traces of power consumption from BattOr, developers still need to attribute the power consumption to specific software modules and hardware components. I have found that signals embedded in the power measurements may be able to be analyzed to (1) sync the with the logs of the device, and (2) divide up the total power consumption into power consumed by specific peripherals. For sync, the insight is that developers may be able to trigger components (e.g., the camera flash) that will show up in the power measurements as well as the logs, and cross-correlate them to find their time offset. For attributing, the insight is that it is rare for several components change their power mode at exactly the same time. Using the synced logs, a change in power consumption following an event can be attributed to the software module and hardware component related to that event. I propose evaluating the accuracy and effectiveness of attributing power measurements in this fashion. I will evaluate the effectiveness of the tools by making these measurements available to the thousands of developers in Google's Chrome team.

Once developers find the modules of their software that are inefficient, it is possible that they will realize the source of the inefficiency is the structure of their program. I am particularly interested in studying the inefficiencies of signal processing algorithms implemented in today's wireless communication devices. Signal processing chains are implemented modularly. This makes design simpler for teams of signal processing specialists to collaborate, because each team can focus on improving the performance of their own block. However, this modular implementation may also sacrifice energy efficiency. One instance of this is in the implementation of a demodulator in a wireless receiver. Demodulators perform bit slicing on symbols that have been channel-equalized. Demodulators could be made more efficient by equalizing the slicing matrix instead of the symbols. This reduces the number of times that the channel equalizer must be run to be once for each channel estimate, rather than once per symbol. I will prototype and evaluate energy-efficiency improvements signal processing implementations using the Atomix **[1]** framework.

Power measurements can be crowd-sourced from built-in power monitors, but many developers do not realize that these power measurements can reveal private information about users. Inferring private information from these power measurements is challenging—today's built-in power monitors operate at a very low sample rate (approximately 100 Hz). However, my colleagues and I have found that an attacker can determine the location of a user using just this coarse-grained power data **[7]**. I propose exploring what other private information might be leaked through these pervasive coarse-grained power monitors as well as fine-grained power monitors that may appear in the future. I will also investigate how to fuzz these data while maintaining their utility to developers.

References

[1] M. Bansal, **A. Schulman**, and S. Katti. Atomix: A framework for deploying signal processing applications on wireless infrastructure. In *USENIX NSDI (Networked Systems Design and Implementation)*, 2015.

[2] **A. Schulman**, N. Spring, V. Navda, R. Ramjee, P. Deshpande, C. Grunewald, V. N. Padmanabhan, and K. Jain. Bartendr: A practical approach to energy-aware cellular data scheduling. In *ACM MobiCom (Conference on Mobile Computing and Networking)*, 2010.

[3] B. Han, **A. Schulman**, N. Spring, B. Bhattacharjee, F. Gringoli, L. Nava, L. Ji, S. Lee, and R. Miller. Maranello: Practical partial packet recovery for 802.11. In *USENIX NSDI (Networked Systems Design and Implementation)*, 2010.

[4] **A. Schulman**, D. Levin, and N. Spring. RevCast: Fast, private certificate revocation over FM radio. In *ACM CCS (Conference on Computer and Communication Security)*, 2014.

[5] L. Zhang, D. Choffnes, T. Dumitras, D. Levin, A. Mislove, **A. Schulman**, and C. Wilson. Analysis of SSL certificate reissues and revocations in the wake of Heartbleed. In *ACM IMC (Internet Measurement Conference)*, 2014.

[6] Y. Liu, W. Tome, L. Zhang, D. Choffnes, D. Levin, B. Maggs, A. Mislove, **A. Schulman**, and C. Wilson. An end-to-end measurement of certificate revocation in the web's PKI. In *ACM IMC (Internet Measurement Conference)*, 2015.

[7] Y. Michalevsky, G. Nakibly, **A. Schulman**, and D. Boneh. PowerSpy: Location tracking using mobile device power analysis. In *USENIX Security*, 2015.