My distributed systems professor, Jim Waldo, once described the construction of computer systems to me as a dark art: one makes the appropriate incantations over arcane, complex hardware and computers become animated, useful machines. Systems take immensely complicated chunks of silicon and make them usable through the power of abstraction and good design. Though still an apprentice in this art, my experience thus far has taught me the thrill and reward of systems building, design, and research.

I came to Harvard with limited understanding of Computer Science as a field but quickly became excited about software systems. I was surprised by how much I enjoyed my Computer Science coursework my Freshman fall and was perhaps most surprised by my newfound ability to leverage abstraction to build structured, modular, and somewhat useful tools. Continuing the next spring, I jumped into a systems programming course, unraveling the software stack further, peeking below the abstraction layer of programming languages and looking at the instruction set architecture, the operating system, and the low-level, gritty details of why we interact with our computers the way we do, and why they work in detail. During Professor Margo Seltzer's course on Operating Systems my sophomore spring, I became enamored with these large-scale systems.

I spent my sophomore summer as an intern at Microsoft in Redmond in order to learn more about life in the software industry. I worked on a team designing new tools for storing enterprise data metadata models on top of SQL server and experienced team-driven development, professional coding practices, and the realities of modern software development. While I enjoyed my work, most formative for me were several discussions I had with with my co-workers and other product teams regarding their thoughts about the future of computer systems. Underlying many of these discussions was a belief that academia was currently "behind" industry, lacking impact on "real-world" technologies.

When I returned to school, I wanted to verify or disprove these claims for myself and I subsequently enrolled in Professor Seltzer's graduate seminar on operating systems. This was the most influential decision I've made in my undergraduate career, aside from making my entry into the field freshman year. I sought counterexamples, that the academic computer systems community was indeed answering questions relevant to modern computing, and found them both in historical contexts, from the THE system to Disco and VMWare, to ongoing projects like Barrelfish and Klee. Not only had computer scientists historically influenced modern computing in profound ways, but many problem domains from the "cloud" to multi- and many-core architectures to program verification were wide open in terms of research potential. While Rob Pike believes, much like some of the detractors I talked with in industry, that *Software Systems Research is Irrelevant*, I discovered real systems building, and began to believe in the potential and freedom of academic research. The continued presence of industry giants like Google and VMWare at top-tier academic conferences like OSDI and SOSP taught me that, at least in parts of industry, academic research and community discussion is meaningful and valued.

Concurrently, I began research with several faculty members. Not only were people in the community building interesting systems, I could as well, and there were several opportunities at my doorstep. I joined my advisor, Professor Matt Welsh, in building resource management systems for swarms of micro aerial vehicles on the RoboBees project (essentially a "cloud with wings"). I spent the year hacking hardware, both writing my first device drivers for TinyOS and making an entry into cyber-physical systems. I would continue this work over the summer, also joining several graduate students and a post-doc in designing a swarm operating system. As part of Professor Seltzer's class, began work on operating system-directed software-assisted microarchitecture,

reducing thermal load in a data center context. I learned about real-world kernel hacking and was exposed to the world of computer architecture, where I was joined by Professor David Brooks and Vijay Reddi, a graduate student. As an offshoot of a class project for my artificial intelligence class, I worked with Professor Radhika Nagpal and Dr. Justin Werfel on bee foraging algorithms.

These research experiences taught me several lessons. I engaged in several simultaneous projects, and research quickly became my primary activity at school, and I found the thrill of experimentation, analysis, and discovery immensely rewarding. I was excited by my projects, and believed in both their novelty and applicability. I learned not only how to evaluate research, but to perform my own and to communicate my findings through writing. I also learned the personal satisfaction of validating the applicability of our work. When our bee foraging study was accepted to an AI conference, I decided to actualize my commitment to having applicable research, and sent our manuscript to the bee biologists who inspired our experiments. Their feedback was positive, and they encouraged us to submit an extended study to a Biology journal as well. Most importantly, I felt that I had made a step towards making our work more broadly applicable; I appreciated the conference's strong reception of our work, but the external validation by scientists whose research is directly impacted by our results was perhaps even more gratifying. I experienced the application of computing to scientific domain firsthand and greatly valued the feedback.

Through this experience, I have become a strong proponent of moving my applied research beyond the academic community and into the real world, whether through industrial contact, commercialization, or outreach to other communities. This is undoubtedly difficult, but, as an applied scientist working on long-term or high-risk, high-reward problems, I firmly believe in actually *applying* my ideas to the real world through active engagement. The benefits are two-way; not only can I benefit practitioners and domain scientists through my research, but I can also learn from them. I hope to do this with my other research projects as well, whether by talking to companies about their data center thermal management, by using RoboBees in real pollination or search and rescue tasks, or by applying our macroprogramming system to other problems of parallelism.

I plan to continue pursuing interesting and applicable research challenges in graduate school and eventually in academia as a professor. I embrace the freedom of academia affords in allowing next-generation, relatively unconstrained but still applicable research and am excited about building systems that allow us to better utilize hardware and explore the abstraction boundaries between applications, systems, and architecture. While my research topics have been been broadly distributed, I have been attracted to the interface between hardware and software; just as distributed systems are necessarily aware of and exploit the network, I am interested in systems that do the same for general-purpose hardware. I am not particularly committed to a particular sub-field of systems research, but as many-core and heterogeneous architectures become prevalent and processor reliability becomes a constraint, there will be many exciting systems challenges to solve. Our ability to embrace new technology and hardware architectures across almost every computing domain, whether in the consumer sector, scientific computing, or cloud operations, is dependent on our ability to meaningfully express and manage computational demands and resources. An NSF fellowship would grant me the freedom to explore potential inter-domain systems research across traditional abstraction boundaries from programming languages to hardware as well as apply computing systems to non-traditional domains via accelerated software and hardware architectures for domain scientists. After the completion of my Ph.D., I hope to pursue a junior faculty position where I can both continue research and teach future generations of computer scientists.