Teaching Statement

Justine Sherry

The opportunity to work with students is a major part of why I am choosing an academic career in a university setting. I enjoy teaching students, find helping students to advance in their careers to be morally fulfilling, and believe that working with student engineers can lead to novel thinking and great research. Most importantly, I believe in the power of computer science and engineering to solve real world problems and I love empowering students to use the computer science toolkit to build great things. In this document I outline my qualifications for classroom teaching, experiences mentoring students in research, and teaching approach.

Classroom Teaching

Experience. I have experience as a teaching assistant in undergraduate networking, and in introductory programming covering all CS1 material and some CS2 material. As an undergraduate at the University of Washington, I was head teaching assistant for Programming I, a role which involved not only teaching students, but guiding other teaching assistants in grading and how to teach their own sections. As a graduate student at Berkeley, I was a teaching assistant for undergraduate networks where I taught sections and designed a course project on routing protocols. These experiences gave me hands-on exposure to the work required to teach at the undergraduate level.

Courses I Can Teach. I am most qualified to teach undergraduate and graduate networking classes. I can also teach most introductory computer science (e.g. programming and data structures) as well as basic systems and security material.

I am particularly interested in teaching two special topics courses related to my research. The first is an academic graduate seminar (focused on paper reading) discussing middlebox processing, network functions, and recent advances in cloud and ISP installations. An academic course like this can help other researchers (including senior graduate students and faculty) get up to speed on the many challenges in this space and help identify ground for collaboration.

The second is a hands-on course targeting both senior undergraduates and junior graduate students. As I mentioned in my research statement, software packet processing allows developers to write new network features in software and deploy them in the network just as easily as if they were deploying on EC2 or a local server. In this hands-on course, students would design, implement, and evaluate new networking services implemented in software. Projects could involve re-implementing common middleboxes from scratch (e.g., SSL proxying or intrusion detection), or in designing new services they imagine. I had the idea for this course because younger students approached me about ideas for new services. I have spoken with two students about using middleboxes to monitor traffic from Internet of Things devices, allowing a user to effectively 'audit' the material that their devices share about them. Another student spoke to me about using a middlebox as a personalized Netflix cache which would pre-fetch episodes of a series the user is watching and hence avoid poor performance at 'peak' hours of usage when the network is overloaded. Neither of these ideas are my own – they came from clever student developers when given the tool of software packet processing to solve problems they cared about. A hands-on course like this both gives students solid engineering experience (which they need for jobs or graduate school), and helps identify potential undergraduate and graduate students to work with in research.

Research Mentoring

I have worked with five undergraduate research assistants in graduate school. The most surprising lesson for me from my mentoring experiences was just how different each student was! Working one on one with students drew my attention to their different skill sets, interests, and goals; together with my students I designed projects that both fit in to my research agenda and helped the students achieve their own goals.

Two exemplar students I worked with were Soumya Basu and Daniel Kim. When Soumya began working with me he already had substantial class and research experience and was interested in graduate school. We quickly integrated him in to the FTMB [1] team and he performed much of the evaluation for our SIGCOMM 2015 paper. Beyond implementation, however, he joined in on research meetings and actively participated in discussions about the core FTMB algorithms. In working with us he took on a creative, graduate-like role. He was accepted to many top schools and is currently in the PhD program at Cornell.

Another student, Daniel Kim, was entirely different – but just as much a success. Daniel got involved in research when he was enrolled in undergraduate networks. Unlike Soumya, Daniel was interested in a career in industry. He had enjoyed networking and wanted more experience to pursue a career in that direction. I assigned Daniel engineering tasks from my active research that tied to open-source, widely used networking code. When Daniel applied for interviews at networking companies, he impressed them with his hands-on knowledge of protocols and networking software. He was hired by Symantec to work on their firewall and intrusion detection middleboxes, focusing on the transition from IPv4 to IPv6.

Overall, I think both students reflect my strategy for research mentoring: to understand the students interests and goals, understand where they align with my projects and goals, and see how we can pursue our goals together.

Teaching Approach

In both classroom teaching and research mentoring, my teaching strategy emphasizes developing clear and effective technical communication skills. A strong engineer must be able to both describe the mechanism of his or her system, as well as explain why it is relevant to problems in the real world. These skills make engineers (and researchers!) stronger for several reasons.

Communication helps students understand. In 'rubber duck debugging', a developer describes a bug in their code out loud to a rubber duck. Often, the mere process of describing the problem out loud enables the developer to discover the source of error. Similarly, students asked to explain technical material often self-diagnose gaps in their understanding or incorrect ideas, which they can then improve on.

Communication helps product teams collaborate. Major engineering projects always involve teams. Interactions between team members involve discussing designs, reporting bugs to other members and explaining algorithms or possible approaches. Ineffective communication delays engineering process or may even lead to poor design when developers misunderstand each other.

Communication helps engineers understand users. All systems are built to solve a problem, and all of these problems reach back to a human need. In my research on middleboxes, I started by asking network administrators what their needs were. In my industry internships, I spoke with marketing and sales teams about what they needed from tools I was building. Students must understand that engineering useful systems necessarily entails talking to the humans who will use them.

Communication helps users understand engineers. The pop-culture image of the scientist or engineer is a lone genius who speaks in a stream of unintelligible jargon. The end result is that average technology users are often mystified by the inner workings of their own computers, and are hence unable to diagnose their own problems with technology. Further, young people exposed to this image are unlikely to envision themselves in a career in technology, instead reserving computer science degrees in their minds for special geniuses, rather than for anyone who chooses to learn. Students who can explain technology make technology more accessible to outsiders

For all of these reasons, I think an important part of the teaching process is helping students learn to describe technology to others.

When I started working with undergraduate researchers, I was always impressed with my students' coding skills, but sometimes was disappointed when my students struggled to explain why they made the design decisions they did or what the purpose of their project was. In research mentoring, we spend significant time teaching students to write and speak on technical content. However, we do not do enough in classrooms. Since these skills are just as critical in industry as they are in research, I think we need to challenge students to be effective technical communicators in classes as well.

One important step is to incorporate communication into exams and projects. On her final exam for undergraduate networking, my advisor Sylvia Ratnasamy always asks students to describe 'what happens' when a user loads a web page. Students must enumerate in writing the Internet components involved and describe how they fit together. Questions that require students to describe how systems work and why are more difficult to grade than multiple-choice

or numeric answer questions, but the extra effort from a few such questions is worth it because written questions force students to practice and improve their ability to communicate technical ideas.

In class projects, requiring students to produce a brief design spec *before* they write their code can serve two purposes. First, it provides practice in communication. I also hypothesize that it will help students to more efficiently implement their projects. Recent CS education research at Berkeley has shown that for small class projects, having students answer questions about what they *expect* their code to do before allowing the students to *implement and test* their code increases the likelihood of students completing the project independently, without instructor help [2].

Another important step in incorporating communication skills in the classroom is encouraging outreach. As an undergraduate in Spanish, I was encouraged to provide translation services at the local community center; as an undergraduate in International Relations I was offered extra credit to volunteer at community events when political leaders spoke in town. I think we can adapt this to the CS environment. I intend to offer a small amount of extra credit to students who participate in 'outreach' activities such as demoing at CS education day, volunteering to provide tech support, or editing technical Wikipedia articles. These outreach activities teach students to discuss technical ideas with outsiders without a strong technical vocabulary; the very audience for the software they will build in their careers.

Conclusion

I am excited for all aspects of being a professor including research, teaching courses, and mentoring students. I enjoy helping students reach their goals, and I think teaching students strong engineering and communication skills will help them use computer science to solve problems that they find valuable.

References

[1] J. Sherry, P. X. Gao, S. Basu, A. Panda, A. Krishnamurthy, C. Maciocco, M. Manesh, J. Martins, S. Ratnasamy, L. Rizzo, and S. Shenker, "Rollback-Recovery for Middleboxes," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, (New York, NY, USA), pp. 227–240, ACM, 2015.

[2] S. Basu, A. Wu, B. Hou, and J. DeNero, "Problems Before Solutions," in *ACM Learning at Scale*, 2005.