Teaching Statement

That instant when something clicks in a person's mind is a particularly rewarding moment for me. I view teaching as an opportunity to expand other people's knowledge and also enhance my own. In exchange for this privilege, I fill my lectures with historic context, metaphors and humor designed to provoke interest and to educate. I am proud that students have on several occasions called me an excellent teacher.

Experience at Reykjavik University

The School of Computer Science at Reykjavik University (RU) is dominated by undergraduates ($\approx$ 800) so most courses are taught at that level. Since arriving in 2011, I have revamped the entire systems, computer architecture and networking curricula. I redesigned the courses on *Operating Systems*, *Networking* and *Computer Architecture* including all material and assignments, taking the ACM "Ironman" standard into account. Further, I created an original course on *Offensive Computer Security* from scratch, as well as a course on the *Structure of Information Networks* based on the emerging field of Network Science. I have received high teaching evaluation scores for every course I have taught.

- *Structure of Information Networks*: 4.4/5 (Summer 2011, 20 students)
- *Operating Systems*: 4.6/5 (Spring 2012, 100 students), 4.6/5 (Spring 2013, 160 students)
- *Offensive Computer Security*: 4.7/5 (Spring 2012, 20 students), N/A (Spring 2013, 28 students)

In the spring of 2012, I was voted the "Teacher of the Year" by the Computer Science Student Union at RU.

Experience at Cornell University

Before working as an Assistant Professor, I actively sought out teaching opportunities during my studies because I found them to be both fascinating and stimulating. I was a teaching assistant (TA) as an undergraduate as well in five courses at Cornell University, including *Discrete Mathematics* (Cornell CS280, Fall 2005, Prof. Jon Kleinberg) for which I received a TA-award, *Graduate Analysis of Algorithms* (Cornell CS681, Fall 2006, Prof. Èva Tardos), and *Data Structures and Functional Programming* (Cornell CS312, Fall 2007, Prof. Radu Rugina) where I taught half the lectures.

At Cornell, I also independently designed and instructed a 4-week course on *Advanced UNIX Tools* (Cornell CS214, Spring 2008) and devised the entire curriculum from scratch. Of the 18 out of 20 students who responded to an anonymous evaluation, all agreed that the course had met or exceeded their expectations and provided glowing reviews.

Philosophy

I believe that good teachers are made, not born. Therefore, I continually seek out new ways to improve my teaching and systematically implement those ideas. Here are several methods and concepts that I have used or plan to use in my teaching.

Engagement through competition

I use contests as a tool to motivate and engage my students, ranging from individual in-class races with real-time status boards to week-long collaborative challenges. I find the volume of material students absorb during these sprints to be quite remarkable, and I detect the positive effect it has on their learning.

I have had long experience with the competition format. At Cornell, I devised an in-class competition for solving tricky problems with short shell-scripts in my *Advanced UNIX Tools* course. I also helped set up a week-long competition for open-ended projects in *Data Structures and Algorithms*, and to evaluate AI for a simple game in *Data Structures and Functional Programming*.

In-course competitions. At Reykjavik University, three of the major assignments in my *Operating Systems and Networking* course had interactive scoring and an associated competition. About a quarter of the class (25/100 students) put in incredibly long hours to get top scores on every single challenge, far beyond the expectations for a top grade. To recognize the effort, I awarded the top students with custom certificates during class, an event I still hear discussed. There is an inherent risk that competition overwhelms those that lag behind on the material. By allowing students to submit anonymous entries to the scoreboards, I was told by a large group they felt they had an opportunity to learn at

their own pace. The final exam also indicated that a large majority of the students had invested significant time on the material that had an associated competition.

Hacking competitions. I further designed an annual "Hacking Contest" in 2011 to raise awareness of computer security which has been held three times now. Students are given source code to services running on a dedicated computer, and raced to exploit the vulnerabilities in these programs. The response was thrilling: over 50 students managed to exploit some vulnerability each year, and many of them had only picked up the skills because of the competition. To determine a winner among the finalists, they were given virtual machines and 30 minutes to collect points for hacking into the machines of one another. These events became a media sensation in Iceland, covered by all major news outlets in 2011, as well as attracting an audience of over 500 people in 2012. In 2013, the contest was a hit on Hackernews and two finalists flew into a 200-person auditorium that was a raving success. A short documentary on the contest was shown for an audience of over 200 people, and the US news agency Vocativ interviewed us as a result (see `http://www.vocativ.com/video/ice-phishing/`.

Idea competitions. I also created an annual "Idea Contest" as a platform for undergraduates to use their creativity to tackle the world's problems. The students could submit new ideas, and vote and comment on existing ones. In last year's contest, the winners were determined by a committee and given a chance to work with faculty members to help their ideas come to fruition. Two teams are now running start-up companies based on the ideas, one is pursuing a PhD to advance his idea about the localization of sound for people with hearing deficiency, and one that developed a game to educate kids about renewable energy placed 4th-5th in Game Design (top 1%) at the international Microsoft Imagine Cup for their efforts.

Providing context

I feel that one should spend the same energy on explaining the context, history and importance of topics covered in class as one motivates one's work at conferences and in grant proposals. Why did someone work on this problem? How is the contribution significant? Why is it useful? What are issues with other solutions? The topics that most interested me and my fellow classmates during our studies were those where the teacher gave a substantial and thought-provoking introduction. The curiosity and excitement that spawned particular problems and solutions in computer science is contagious if only the teacher conveys it properly. I have yet to encounter a topic that warrants being taught and does not have a gripping back story.

When I teach any topic, be it on x86 assembly, the UNIX environment or network protocols, I attempt o present the material in a way that allows students to understand the sequence of ideas and logic that culminated in the state of the art, including brief historic accounts of how the solution evolved. In class, I motivate every problem by a use case and then show the students the ropes by explaining the thought process needed to solve it interactively. We start off with the simplest possible attempt at a solution, and then gradually add complexity while justifying every step on the way. Ultimately, I try to dedicate enough time in my courses to provide context so that all of my students understand *why* we are studying the topic or problem, and *why* it is cool.

Teaching systems courses.

The world of systems is enormous, full of special cases and legacy protocols, which students can find unwieldy. The constructions are invariably complex, and the vocabulary dense. Unsurprisingly, I have found that students who do not delve deep into the new concepts early in the course tend to fall behind and give up. Therefore, it is most effective if students regularly read the material and absorb the concepts right from the start. The most effective technique I have found that encourage such behavior is to ask each student to submit before each lecture:

- their own brief synopsis of the paper or concept;
- the key benefits of the contribution, and
- the drawbacks of the approach.

These summaries count towards the final grade, so students have an incentive to understand the concepts in a timely fashion. I had a successful trial of this technique in an undergraduate course, and intend to make this a standard in a future graduate course or seminar.

Furthermore, I find it important to that students get hands-on experience with real systems. In my courses, I structure the assignments so that students all get practical experience with the major topics of the course, including the use of various operating systems and common programming languages. To name some examples, students learning about networking protocols do assignments on Cisco emulators. Students learning x86 assembly will decipher unknown

binary code compiled from C on Linux. Students learning about information networks will implement algorithms in Python and evaluate on real-world graphs. The same applies to networked systems: I want the students in my courses to get cracking early on environments like PlanetLab and Amazon's EC2 to become familiar with the "real world". In addition to using these environments in my teaching, I set up a 100-node cluster intended to be a playground for students to become familiar with "real-world" networks and to try out new ideas. Hands-on learning experience can turn a systems course into an exploration rather than a dry memorization of concepts.

Providing incentives to students

With the current structure of undergraduate studies, I have found that students lose long-term visibility and spend their time predominantly on activities that carry short-term benefits. Many students regularly defer assignments until just before a deadline, or ignore events and activities that are in their best interest but not compulsory. I have recently worked to create incentive mechanisms to help students even better invest in their own future.

Battling procrastination. Students are notorious for procrastination. The students who performed poorly in my classes tended to start their assignments and exam preparation late, and often had other organizational issues. Breaking this habit can be very important for the person. To aid this, one mechanism I have encountered to encourage early submissions is a kind of a descending-bid auction: reward students with extra credits depending on how early they submit the homework before the deadline. I am planning to experiment with this this approach in some of my classes in the spring.

Improving attendance. Another technique to encourage students to show up for events and classes that they skip is to create systematic incentives. Together with a team of students, we have created a prototype of a virtual currency called "RU-bits". At the beginning of a lecture or event, a bespoke QR-code is shown on screen which, when scanned, gives attendees a number of credits in their own account. The idea is for students and departments to compete for points, with rewards being given out periodically in proportion to the number of credits accrued. In addition, students can use the credits to bid on priority access to lockers and student meeting rooms that are in short supply at my university. To keep a focus on positive reinforcement, we deliberately refrain from translating credits into something with direct monetary value, such as printing quota. I am planning to deploy the RU-bits system in the spring of 2014.

Competencies

In addition to its other qualities, I believe teaching is a means to find and evaluate students to work with in the future. This is an important point for me, since the small graduate program at RU has limited the growth of my research group. The prospect of interacting with the next generation of researchers makes me excited to teach a graduate-level course or seminar on *Cloud Computing* or *Networked Systems* (such as caching systems, peer-to-peer protocols and content distribution networks) early in my appointment. I would emphasize on covering papers on seminal discoveries and recent advances in the field, and with the additional goal of identifying new ideas to advance the state-of-the-art technologies.

At the undergraduate level, I would be happy to teach courses relating to *Networking and Cloud Computing*, *Computer Architecture*, *Operating Systems*, *Information Networks and Social Networks* as well as general introductory courses. Furthermore, I would be thrilled to offer my new course on *Offensive Computer Security* which has been incredibly popular. I am also comfortable with teaching courses on *Discrete Mathematics* and the *Analysis of Algorithms*.