Educating the Next Generation of Spammers

Joel Sommers
Colgate University
Hamilton, NY, USA
jsommers@colgate.edu

ABSTRACT

Compelling experiences in introductory courses make a key difference in whether non-majors develop an interest in computer science, possibly even converting them into undergraduate majors or minors. In this paper we advocate integrated hands-on laboratory style activities to provide such pivotal experiences. In the lab activities we describe, students do not engage in programming, yet they learn to think computationally by engaging in computational activities. The course in which these labs are implemented is oriented around three aspects of the the internet's *underside*: its techno-scientific underpinnings, environmental and energy problems and promise brought on by its rapid growth, and security threats associated with its use. We describe the goals and content of the lab activities, as well as various challenges encountered through their implementation. We also discuss student responses and future directions.

Categories and Subject Descriptors

K.3.2 [**Computer and Information Science Education**]: Computer science education; C.2.5 [**Local and Wide-Area Networks**]: Internet (*e.g.*, TCP/IP)

General Terms

Experimentation, Measurement

Keywords

Non-Majors, Computational Thinking, Active Learning, Problem-based Learning

1. INTRODUCTION

Introductory courses play a pivotal role in whether students develop an interest for continuing to take courses within a discipline, especially non-majors with no prior experience. Early student experiences can invigorate an existing interest, open up a new one, or, alas, cause a student to seek intellectual stimulation elsewhere. Departments in institutions with core curriculum requirements (*e.g.*, liberal arts colleges) are often presented with golden opportunities each semester: through curiosity, the fate of scheduling, or acts of serendipity, students enter our classrooms. While they are there ostensibly to fulfill a requirement, the favorable circumstance presented to us is simply that *they are there*. In computer science these opportunities are not insignificant [11, 22].

In this paper, we describe a course for non-majors that was designed to introduce students to scientific practices from the perspective of computer science. The course is oriented around the internet (the author's own research area), and follows a theme of exploring *undersides* of the internet: the technical and scientific underpinnings of the modern internet; environmental problems and social challenges brought on by the internet's rapid growth as well as promising efforts to mitigate these problems; and the many security challenges facing the internet— botnets, denial of service attacks, worms, viruses, spam, etc.—as well as the responses by the scientific and operations community to try to address these threats. In this context, students learn what sorts of questions can be addressed through scientific means, the nature of scientific inquiry and debate, and how scientific practices actually work on the ground. A primary method by which this is accomplished is through in-class laboratory sessions. Through these activities, students learn about computer science by *doing it*. Indeed, the key goal for these sessions is for students to learn to approach problem solving from a computational perspective [14, 23].

One of the main challenges with teaching and learning computer science, especially with non-majors (and the potentially computer-phobic) is that much of our subject matter is rather abstract. The first time the term "packet" is introduced to students, there is visible consternation: the internet is modern magic to most students. It can be hard to picture a packet, or a path that a packet might take through the internet, or what the internet looks like (indeed, characterizing the topology of the internet has been a source of compelling research questions for some time). In-class laboratory exercises provide a way to complement the conceptual knowledge that students gain through readings, discussion, and lecture. Lab exercises also provide a focal point for addressing issues like scientific methodology, designing an experiment, measurement instrument quality and data quality, data gathering, hypothesis testing, and many other issues that come up when performing experiments or when thinking about the process of science.

In his book *Representing and Intervening*, the philosopher of science Ian Hacking introduces the concept of scientific realism. He relates an anecdote of a physicist friend who describes an experiment in which the friend "sprays" electrons on an object. Leading up to this description, Hacking discusses whether an electron is "real" or merely a convenient theoretical construction. He then states: "So far as I'm concerned, if you can spray them then they're

SIGCSE'10, March 10–13, 2010, Milwaukee, Wisconsin, USA.

real" [16]. His point aptly describes why hands-on exercises are critical for students in introductory computer science courses: until students can *do* something with their knowledge, that knowledge remains lifeless and somehow unreal.

An important design decision for the labs is that there is no direct programming involved. Instead, students work in pseudocode and bring other resources to bear (*e.g.*, a combination of net-accessible and custom software tools provided to the students) when investigating particular phenomena and when attempting to formulate questions and hypotheses. As such, our approach differs from more conventional CS0-style courses that emphasize aspects of computer literacy or that introduce students to a selection of topics in computer science through simple programming, *e.g.*, in javascript.

While our hands-on approach presents interesting learning opportunities, there are some key challenges as well. In particular, there are logistical challenges involved in designing substantive laboratory activities that can be accomplished within the confines of a standard 50 minute class period. Moreover, set up of laboratory hardware and software demands technical resources, system administration expertise, and time. Even for a class of modest size, the setup overhead costs can be significant. Our goal is to release our tools and materials to the community so that others may take advantage of and build upon our approach.

As part of our effort we have developed several laboratory activities, six of which are described in this paper. Qualitative data have been collected thus far on student experiences and whether students feel that the lab exercises contribute to their understanding of course material. We are encouraged by the fact that student feedback has been overwhelmingly positive. Indeed, several students have since enrolled in other computer science courses. We believe, moreover, that our course structure could serve as a model for institutions with similar curricular requirements.

The rest of this paper is organized as follows. In Section 2 we describe the laboratory activities and discuss challenges associated with their implementation. We follow in Section 3 by describing student experiences and feedback. Section 4 presents work related to ours. We conclude and consider future directions in Section 5.

2. AN OVERVIEW OF THE LABS

While each lab is designed to enable students to gain hands-on experience with concepts and ideas from readings, lecture, and class discussion, there are also a number of learning goals that transcend any particular lab. These goals are aligned with the larger context of the course, which is to introduce students to the methods and practices of science through a particular disciplinary lens. In particular, one goal is for students to gain experience in carefully collecting and analyzing data. Moreover, since many kinds of internet data tend to be noisy, a related goal is for students to learn to assess data quality and the validity of a measurement. Such an ability can be challenging to attain, as it can require deep domain knowledge and/or a full understanding of a measurement tool and the assumptions built into it. Our position is that although non-majors are not likely to fully grasp all the issues at stake (indeed, these are difficult for *any* non-expert to understand), they are capable of learning enough to appreciate the importance of evaluating the quality of an experiment's outcome. Through this, it is hoped that they will learn to see science not as a black box that yields answers, but as an imperfect yet powerful process by which to understand the world. A goal that follows from the problem of evaluating measurement data is for students to be able to analyze the design of an experiment and to suggest ways in which an experiment might be improved given limitations of the instruments or data at hand, or a refinement of the question of interest. Finally, we note that although our labs are designed within a particular academic context, we view our approach as well as the specific content of the labs to be more widely applicable.

For each lab, students work in pairs. In some of the labs, there are numbers, symbols, and equations. Having students work in pairs helps to ease the problem of confronting these scary creatures. After performing the hands-on part of each lab, students work through a number of synthesis-type questions which essentially ask students to make sense of what they did in the lab activity and to connect ideas from class with the labwork. Sometimes these questions were completed as part of the lab activity, sometimes they were assigned as homework problems, and in a few instances these questions appeared as exam questions.

2.1 Brief laboratory descriptions

2.1.1 What does the internet look like?

One of the basic definitions of the internet that is introduced early in this course is that *the internet is a network of networks*. This is to say that the internet is composed of many separate, administratively independent networks. The collection of these networks and the connections among them are what forms the internet. In this lab, students use the well-worn internet measurement tool `traceroute` (more specifically, the `paris-traceroute` variant [7]) to learn more about the topology of the internet and how the internet routing algorithms result in specific paths that packets take through the internet.

Students choose two separate remote hosts, one in the US, and one outside of the US, and use the `traceroute` tool to each of these destinations. The non-US destination must be a looking glass `traceroute` server [4] so that students can evaluate two directions of a route. They then analyze the data from `traceroute`, as well as gather some additional data based on the `traceroute` output, such as the estimated geographic location of each router and the autonomous system to which each router belongs. There are two interesting things about `traceroute` that students are made aware of: first, that it is one of the only methods that the research and network operations community has for discovering the path that a packet takes through the internet; and two, that there are many limitations and pitfalls associated with its use. While `traceroute` is designed to give a hop-by-hop listing of the routers through which a packet passes on its way from a sender to a receiver, the data produced can be very noisy, confounding interpretation of the output.

For the purposes of the lab, though, the noisy data is useful. Prior to the lab, we discuss in class some of the reasons behind why `traceroute` data can produce difficult-to-interpret results. For example students learn about packet loss, packet filtering, and challenges presented by the fact that network providers can configure routers in their networks in ways that limit the usefulness of `traceroute`. Thus, one of the key learning goals for this lab is for students to critically analyze the data at hand and to evaluate its validity and meaning.

2.1.2 Spam, spam, spam

First, let us be clear: the lab that inspired title of this paper actually has to do with *stopping spam*, not facilitating its spread. One of the main goals for this lab is for the students to learn how a modern spam filter works, and more generally to learn to think algorithmically about how something like a spam filter can be implemented in a computer. Interestingly, some students think that since they do not receive much spam in their email inbox, it is no longer a se-

rious problem. In reality, over 90% of all email on the internet is unsolicited bulk mail. Clearly, spam continues to be a significant problem.

The main activity in this lab is for students to learn how a widely used spam filtering engine called SpamAssassin [3] works. At the heart of SpamAssassin is a large set of rules that are matched against every incoming email to a mail server. These rules test various parts of an email for known characteristics of spam, or for characteristics that suggest that an email is not spam. These rules contain scores that collectively contribute to an email's overall score. A simple threshold is used to classify an email as spam or not. Another key algorithmic component of SpamAssassin is a machine learning algorithm that enables the system to adapt to evolving trends in features of spam and to be tuned to the kinds of email that a given user receives. Thus, students also learn some of the basic ideas behind statistical classification.

For better or worse, the best way to learn the basics of how SpamAssassin works is by creating and sending spam. As part of this lab, students had to construct spam within certain parameters to sell counterfeit watches and prescription painkillers. Their spam was relayed through an anonymous remailer (with built-in checks on the destination address so that students could only send spam to designated addresses) and passed through SpamAssassin. It is important to note that all the spam generated by the students was processed within a (nearly) closed network in order to minimize the possibility of their spam escaping into the rest of campus and the internet. Their goals were to create some messages that would be tagged as spam by SpamAssassin, and some that would not be tagged, and more importantly to investigate *why* in each case.

2.1.3 Web performance prediction

Use of a web browser has become second nature to students. Indeed, students entering college today have hardly known a time when the web did not hold the position as the central resource for information. Although they may be comfortable using the web, many students are acutely aware of how shallow their knowledge is about how the web functions. When something happens that is unexpected — a web page takes a long time to load or does not load at all — the limits of their knowledge become all too apparent.

The ability to make predictions is fundamental to science. Leading up to this lab, students learn about web protocols, general concepts about underlying transport protocols, and the basics of how web pages are constructed. With that knowledge, their task in this lab is to try to evaluate the factors behind what causes some websites to load quickly and others to load more slowly. The end goal of their evaluation is to try to build a prediction model for web performance.

The main tool the students use for gathering data to evaluate performance and build a model is the firebug extension [5] to the firefox web browser. Firebug provides the ability to measure how long different elements of a web page take to download, and how long the entire page takes to download and display. These measurements can be displayed in real time as a page is loaded, making them quite visually compelling for students learning how the web functions. (Note that students are shown how to disable caching in order to evaluate effects due to the network and remote server.)

To build their prediction model, students can use the data collected from firebug, as well as simple round-trip delay measurements using `ping` and topology measurements using `traceroute`. Students can also make use of measurement tools available from the Measurement Lab [1] in order to test network performance between their computer and a measurement lab host near a target website (if one is available). Tools are provided to create simple 2D scatter plots of different combinations of their data; these scatter plots form the basis for their predictions. For some website selections, a clear set of clusters arises in the scatter plot. For others, the patterns are less obvious. In any case, a learning goal of building this model is for students to recognize that they can develop a hypothesis for why a given site (or cluster) exhibits the performance it does, and using the tools at hand to test their conjecture.

2.1.4 Estimating personal power consumption

Electronic devices of all kinds pervade modern existence. Many students regularly use laptops, smartphones, gaming consoles, and various other devices, each of which consumes some amount of electricity. Although some students are keenly aware of their power consumption footprint (even though they might not be able to quantify it), others are not. This lab consists of two parallel components. In one, students keep a journal of their use of high-tech devices over several days. They log how long they use each device and record some basic information about the device itself. The second component begins partway through the time during which students keep their journal. They are asked to bring a variety of devices into class in order to measure their power consumption. At the end of their journalling, they use the measurements taken in class along with their journal entries to form an estimate of their average daily consumption of power due to high-tech devices.

The power measurements are taken using handheld digital multimeters which have the capability to download the measurements to a computer for graphing. The devices brought into class are measured under different usage scenarios, *e.g.*, while playing an intensive game, watching a video, or allowing the system to remain idle. Server-class systems are also brought into class for measurement in order for students to see the kinds of computers that run commonly used cloud services and to motivate the massive amount of power used by internet services and infrastructure.

Once the journalling period is over and an average daily power consumption is computed, students are asked to quantify the amount of power in terms of the number of pounds of coal that would need to be burned in order to support their power needs. For some, making this calculation results in a rather disturbing image, but helps to achieve a goal of bringing an awareness to students that may or may not have been there previously.

2.1.5 Breaking simple ciphers

One of the key security technologies behind why the internet has been successfully harnessed by the business world is cryptography. While students can clearly grasp *why* cryptography is important and what it generally accomplishes, the details of how different cryptographic ciphers work can be challenging to grasp, especially for the math-phobic. Our position is that focussing on the mathematical details in a course designed for non-majors can be counterproductive. On the other hand, working with relatively simple ciphers can help to provide a concrete context for students to understand the terms and basic process of encryption and decryption. Thus, in this lab students learn how simple cryptographic ciphers work, and attempt to decode two ciphertexts given knowledge of the cipher used to encrypt each of them but without knowledge of the keys.

Two ciphertexts are used. One is encrypted using a monoalphabetic subsititution cipher, the second is encrypted using a Vigenere cipher. Prior to the lab session, cryptanalysis of these basic ciphers is discussed in class and students discover ways in which they can be broken. Breaking the Vigenere cipher is the more difficult one of the two. To solve this challenging problem, students are led to consider the fact that if they can estimate the length of the key, then breaking the ciphertext boils down to the simple task of breaking

a series of Caesar ciphers. A standard method for estimating key length is provided as a reference, thus enabling students to break the code.

A web page with some javascript is used to enable students to evaluate letter frequencies and to specify how letters are to be substituted (in the case of the monoalphabetic substitution cipher). The page also enables students to reorganize the ciphertext into a specified number of columns and to perform frequency analysis on each column (these features are useful for breaking the Vigenere cipher).

2.1.6 Scanners and intrusion detection

There are few people who have never experienced having their computer infected with a virus, worm, or other form of malware. For many, such an event motivates questions such as: *Why doesn't operating system vendor X produce a better product? Why is my computer a target of hackers? Why is malware so pervasive?* and *What can I do next time so that my computer doesn't get hosed?* In this lab, students initially play the role of trying to identify vulnerabilities in networked-computers and to understand some of the causes and implications of those potential problems. To do that, they use the Nessus [2] and Nmap [12] tools to scan a set of hosts in a closed, relatively small network, that are known to have vulnerabilities. Using these tools, they are able to discover the set of vulnerabilities that exist in the (mini-)network, and the basic implications of those vulnerabilities.

They then switch roles and try to identify ways in which to mitigate the threats posed by the vulnerabilities that they've identified. A simple and standard defense mechanism that students learn about is a firewall. Through the exercise, students learn how a firewall might be configured to mitigate a vulnerability. Perhaps more importantly, they learn about various shortcomings of firewalls and why they are an ineffective means of defense against a large class of attacks.

One of the goals for this lab is for students to experience firsthand how easy it is for a remote attacker to gain information about a potential target. The motivation for using such powerful and potentially dangerous tools is two-fold: first, to help students recognize the importance of open exchange of information, even if that information could be used for harm, and second to encourage them to configure their own machines in responsible ways.

2.2 Challenges Faced

Of the various challenges posed by instituting in-class labs, probably the most significant is the time constraint. Executing a meaningful laboratory exercise within the confines of a standard 50 minute class period has proved difficult. Indeed, students have also recognized and commented on this problem. While we expect that some of the time constraint problems will be resolved as the labs are refined and revised over time, we also expect that there will continue to be a fundamental tension between providing meaningful and thought-provoking problems to grapple with and overwhelming students with too much to try to accomplish in the time allotted.

For each of these labs, we have spent quite a bit of time setting up or creating tools that are accessible to the students through a standard web browser. While there are many tools that advanced students might be expected to use to accomplish various tasks, our goal was to provide a familiar environment from which to base student activities. Another key logistical challenge has been setting up the laboratory infrastructure, including an isolated network, various servers, student accounts, honeypot machines for vulnerability testing, and the like. Our hope is to generalize and package our configurations where possible so that others can more easily set labs up.

3. EVALUATION

Thus far, qualitative feedback that has been collected from students has been overwhelmingly positive. By these measures, the labs have been very successful and the students have responded very positively to them both in terms of actually enjoying them and feeling like the lab activities contributed to their knowledge base. Below, we provide a representative selection of comments provided by students in response to the following prompt: *Please comment on how the in-class lab exercises contributed to your learning in this course*.

- I really liked the labs. They weren't overly stressful or long, but they did help me to understand the material better because it was a hands on experience with the tools [we] talked about in class.
- I think the labs helped my understanding of the course material because it was a more hands on approach rather than just reading about it.
- The labs definitely helped me understand the material. I learn better when I see things in actions and can see how they work so the labs really helped me.
- Yes...very much so....Although more time should be given in class for lab completion. Maybe make them a 2-day process...
- We actually learn things, instead of watching a teacher flaunt their knowledge.

Finally, although we do not yet have long-term data, our initial experience also suggests that our approach has been instrumental in drawing in students to take further computer science courses, a not insignificant motivation behind our efforts. Approximately 15% of students who have taken this course have subsequently enrolled in a programming-based introductory course. Given the fact that students were enrolled in the course described in this paper ostensibly to fulfill a requirement, we view this as a highly encouraging development.

4. RELATED WORK

There have been many innovative efforts over the years to develop compelling ways to introduce non-majors to computer science. Naturally, our use of in-class laboratory exercises builds on these prior efforts. Moreover, we draw inspiration from problem-based approaches that do not use programming.

More specifically, although the process of design for a non-major's course described in [15] is directed toward a course in which students do programming, their general approach is applicable. In particular, the aspects of topical relevance and choice of a compelling context, and of making the classroom experience a social experience have been underlying goals throughout the development of our laboratory activities. Similarly, our effort is comparable to [19] in which a course for non-majors is designed explicitly around the internet. Indeed, it should not be surprising that many others have noted that the internet, network security, and related areas provide a compelling frame of reference for introductory courses [13, 17, 18].

The problem-based nature of our laboratory activities organized around current events has similarities to the course described by Astrachan [6]. The focus in [6] is for students to ponder larger computational issues rather than focussing on details that they are likely to forget. We believe that an approach that includes meaningful activities in which students can grapple with different problems

in a tangible way can help to create a bridge between low-level details on the one hand, and implications on the other.

While it is widely held that students learn best by doing, how and exactly what students should do in an introductory course for non-majors has been the subject of much debate. Many introductory approaches favor a programming-based experience, but others advocate a non-programming-based approach; recent examples may be found in [10,20,21]. While a programming-based approach may certainly be appropriate for certain institutions, it may not be for others that have unique curricular constraints, *e.g.*, in some liberal arts colleges [8,9,24].

5. CONCLUSIONS AND FUTURE WORK

In this paper we describe in-class lab activities created for a course for non-majors. The labwork is designed to enable students to grapple with scientific problems in the context of different aspects of the internet, and to provide them with tangible experiences with what are often rather abstract entities. While we contend that hands-on labwork can provide an effective complement to other classroom activities, there are two significant challenges that we have faced with this approach: time and setup costs.

In the future, we hope to develop open courseware materials to assist others wishing to adopt such activities. We also intend to investigate ways in which our approach could be scaled to large classes. Finally, a question we will consider is whether a course like ours for non-majors could be developed entirely around hands-on experiences, and how such a course might be orchestrated.

6. REFERENCES

[1] Measurement lab. `http://www.measurementlab.net/`.

[2] Nessus. `http://www.nessus.com/`.

[3] SpamAssassin. `http://spamassassin.apache.org/`.

[4] traceroute.org. `http://www.traceroute.org/`.

[5] Firebug: web development evolved. `http://getfirebug.com/`, 2009.

[6] O. Astrachan. Pander to ponder. In *SIGCSE '09: Proceedings of the 40th ACM technical symposium on Computer science education*, pages 192–196, March 2009.

[7] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira. Avoiding traceroute anomalies with Paris traceroute. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 153–158, October 2006.

[8] A. Brady, P. Cutter, and K. Schultz. Benefits of a CS0 course in liberal arts colleges. *Journal of Computing in Small Colleges*, 20(1):90–97, 2004.

[9] C.D. Cliburn. A CS0 course for the liberal arts. In *SIGCSE '06: Proceedings of the 37th SIGCSE technical symposium on Computer science education*, pages 77–81, 2006.

[10] T. J. Cortina. An Introduction to Computer Science for Non-majors Using Principles of Computation. In *SIGCSE '07: Proceedings of the 38th SIGCSE technical symposium on Computer science education*, pages 218–222, Covington, Kentucky, USA, March 2007.

[11] A. L. Foster. Student interest in computer science plummets. *The Chronicle of Higher Education*, 51(38):A31, May 27, 2005.

[12] Fyodor. Nmap. `http://www.nmap.org/`.

[13] C. Gurwitz. The Internet as a motivating theme in a math/computer core course for nonmajors. In *SIGCSE '98: Proceedings of the twenty-ninth SIGCSE technical symposium on Computer science education*, pages 68–72, 1998.

[14] M. Guzdial. Paving the Way for Computational Thinking. *Communications of the ACM*, 51(8):25–27, August 2008.

[15] M. Guzdial and A. Forte. Design process for a non-majors computing course. In *SIGCSE '05: Proceedings of the 36th SIGCSE technical symposium on Computer science education*, pages 361–365, 2005.

[16] I. Hacking. *Representing and Intervening*. Cambridge University Press, 1983.

[17] A. M. Holland-Minkley. Cyberattacks: A Lab-Based Introduction to Computer Security. In *SIGITE '06: Proceedings of the 7th conference on Information technology education*, pages 39–45, Minneapolis, MN, USA, October 2006.

[18] C.D. Knuckles. A net-centric curricular focus. *Journal of Computing in Small Colleges*, 17(6):75–81, 2002.

[19] S. Kurkovsky. Making computing attractive for non-majors: a course design. *Journal of Computing in Small Colleges*, 22(3):90–97, 2007.

[20] J. Marks, W. Freeman, and H. Leitner. Teaching applied computing without programming: a case-based introductory course for general education. *SIGCSE Bulletin*, 33(1):80–84, 2001.

[21] S. Pollard and J. Forbes. Hands-on labs without computers. In *SIGCSE '03: Proceedings of the 34th SIGCSE technical symposium on Computer science education*, pages 296–300, 2003.

[22] J. Vegso. Low Interest in CS and CE Among Incoming Freshmen. `http://www.cra.org/wp/index.php?p=104`, February 6 2007.

[23] J.M. Wing. Viewpoint—Computational Thinking. *Communications of the ACM*, 49(3):33–35, 2006.

[24] B. Zimmerman. Content and laboratories of a computing science course for non-majors in the 21st century. *Journal of Computing in Small Colleges*, 19(5):68–77, 2004.