HANDS-ON INTERNET WITH SEATTLE AND COMPUTERS FROM ACROSS THE GLOBE[*]

Scott A. Wallace
Washington State University – Vancouver
Vancouver, WA 98686
wallaces@vancouver.wsu.edu

Monzur Muhammad
Polytechnic Institute of New York University
Brooklyn, NY 11201
monzum@cs.washington.edu

Jens Mache
Lewis & Clark College
Portland, OR 97219
jmache@lclark.edu

Justin Cappos
Polytechnic Institute of New York University
Brooklyn, NY 11201
justinc@cs.washington.edu

ABSTRACT

The Internet Connectivity module is a short assignment covering distributed computing and networking. The Internet Connectivity module is part of the curriculum created for the Northwest Distributed Computer Science Department and is built upon the Seattle distributed computing platform. In this paper, we describe the module and illustrate how Seattle facilitates networking projects and experiments that use computers/resources from across the globe. In addition, we describe how the Internet Connectivity module was used in two courses, provide some comments on students' reactions to the project, and conclude with suggestions for faculty considering how to use this module in their future courses.

INTRODUCTION

Designing an innovative course curriculum is a time-consuming and challenging task for faculty anywhere. At smaller institutions, the resources of both faculty and

[*] Copyright © 2011 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

departments are often limited. Yet smaller institutions play a significant role educating over 50% of the computing undergraduates nationwide [2].

"The Northwest Distributed Computer Science Department" (NWDCSD) [3] was formed to help address some of the resource challenges faced by Computer Science faculty at small institutions. This community now involves 19 active participants (those attending 3 or more events) from 14 institutions that have been working together to build and share reusable course projects that are compelling to students and that can be adopted by faculty without a major curricular change.

Over the past three years, the community has developed a number of curriculum modules (16 at the time of writing) that target students in one of two ways. For those already in the major, modules intend to bring interesting and contemporary projects into the classroom with relatively low barriers of entry for instructors. For non-computer science students, modules aim to introduce one or more aspects of computational thinking and to provide students with a better-developed view of the field.

In this paper, we discuss a short (one or two hour) curricular module targeted toward computer science students and focused on Internet connectivity issues. In the following section, we describe the Internet Connectivity module and the Seattle platform upon which it is built. Next, we describe briefly how it was used in two different classes over the last two years and comment on both student and faculty perceptions from those two years of usage. Finally, we conclude with suggestions for future use.

THE INTERNET CONNECTIVITY MODULE & SEATTLE

The Internet is stitched together by diverse network technologies. Due to this diversity, some properties of the Internet simply cannot be observed or simulated with a local-area network (LAN). The Internet Connectivity (IC) module lets the students experience first hand interesting characteristics about the connectivity of the Internet. Participants will become familiar with network properties such as non-transitive connectivity, forwarding and routing, Internet WAN latencies, and NAT (network address translation).

The IC module is part of a larger networking/distributed systems research and educational platform called Seattle [1]. Seattle is a peer-to-peer platform upon which students and researchers can run their own code on end-user devices distributed across the world. Since these are real everyday devices that are used by the community, Seattle users are able to observe real Internet behavior when running software on the Seattle testbed. Users are able to experience the exact behavior that a device on the Internet has rather then just a simulation. Seattle allows users to donate, share and obtain resources in order to run various experiments in a secure and scalable manner. Similar platforms such as PlanetLab [4] provide users high end VM machines spread across the world but they do not provide the level of diversity that Seattle provides. Unlike PlanetLab, which runs on dedicated machines, Seattle can and does run on a diverse set of devices. Specifically, Seattle provides users with nodes that include laptops and mobile phones as well as devices that lay behind NATs. Thus, the systems involved in a Seattle experiment at any given time are likely to encompass many different device types, operating systems and connectivity bandwidths.

In this paper, we focus our attention on the Internet Connectivity module-a short assignment that can be completed in one or two hours. The Internet Connectivity (IC) module walks participants through a series of steps that involve acquiring resources/computers from around the world, executing programs on these machines, and logging information from the runs. The IC module includes a program that students run to create a matrix of ping times between all the machines involved in the experiment. Below, we describe the module in more detail and provide command-line segments showing how some of these tasks are performed with Seattle.

Getting Up And Running With Seattle

Students using Seattle for the first time begin by creating an account on SeattleGENI, a public portal used to manage Seattle resources. From here, students can acquire resources/machines from the network that can be used to run experiments. The portal also maintains a user's public key which is used to ensure the integrity of commands and data sent between machines.

Once resources are acquired through SeattleGENI, students can access these network nodes using a few simple commands within the Seattle Shell (Seash) [5]. Seash is a shell specifically designed to communicate and interact with resources that users have acquired. The following three command lines illustrate the process of: 1) loading a keypair; 2) informing Seash to perform actions with a specified identity; and 3) searching for acquired network resources:

```
!> loadkeys wallaces
!> as wallaces
wallaces@ !> browse
['192.138.213.236:1224', '129.187.143.100:1224',
'204.85.191.10:1224', '128.193.33.8:1224']
Added targets: %1(204.85.191.10:1224:v22),
%2(192.138.213.236:1224:v4), %3(129.187.143.100:1224:v8),
%4(128.193.33.8:1224:v20)
Added group 'browsegood' with 4 targets
```

In the segment above, only the browse command produces output. Here, we see the IP addresses of the four available machines that are now ready to run our programs. The remaining output indicates that we can refer to each machine with a specific alias (%1, %2, %3 and %4) or address all four target machines with the group name "browsegood".

Performing Tasks on Remote Machines

The Seattle platform executes programs written in a restricted Python language called Repy [5]. The Repy programming language was developed to provide a certain amount of security as not to allow any rogue program to access the host system directly. The language also provides a broad set of API calls that allow a beginning user to quickly write networking code. This allows the programmer to focus on concepts without getting lost in the complexity of the socket API. In Seash, executing a Repy program on a remote machine is a simple one line command:

```
wallaces@ !> on %1 run helloworld.repy
wallaces@ !> on %1 show log
Log from '204.85.191.10:1224:v22':
    Hello World
```

The output from the sample program is stored in a log file that we can retrieve and display as in the second command above. In addition to addressing a single machine, the "on" command can be used to address a group by substituting the group name for the IP address or alias. Seattle also allows users to easily kill remote processes using the "stop" command.

Measuring Latency Between A Group of Machines

Next, students examine the ping times between remote machines by running the "allpairsping.repy" program. The program builds (and updates) a matrix of ping times between each machine running the program as in the following table:

	192.138.213.236	**204.85.191.10**	**128.193.33.8**	**129.187.143.100**
192.138.213.236	0.00s	0.17s	0.09s	0.10s
204.85.191.10	0.03s	0.00s	0.09s	0.51s
128.193.33.8	0.09s	0.09s	0.00s	0.19s
129.187.143.100	0.29s	0.11s	0.40s	0.00s

While measuring the connectivity between machines, students may notice a pair of machines that are unable to communicate with each other. This often occurs when two machines are unable to directly communicate with each other due to some network misconfiguration. However there may be an intermediate node that can communicate with both the nodes that are unable to communicate with each other. This is known as non-transitive connectivity. Later parts of the IC module illustrate how to overcome this problem by setting up a packet forwarder so the two nodes that are unable to communicate directly, can now communicate through the intermediate node.

Packet Forwarding , NATs and Beyond

The IC module further teaches the participants about network address translator (NAT) nodes and the problems that users may run into when running a network code on a machine that may be behind a NAT. By the end of the assignment the participant should know about latency, non-transitive connectivity, packet forwarding and NAT nodes.

The Seattle webpage has an educators portal that provides several other assignments in addition to the Internet Connectivity module presented here. These assignments cover more advanced networking concepts such as peer to peer routing and reliable messaging.

TWO YEARS, TWO COURSES

The brevity of the Internet Connectivity module helps to make it a relatively easy addition to a course whose curriculum is already well established. At the same time, the resources provided by the Seattle platform allows students (or instructors) to pursue more advanced distributed computing assignments over the course of multiple weeks.

In 2009, one of us (Mache) offered an information security course to a class of 7 undergraduates (mostly juniors and seniors) at Lewis & Clark College. The instructor devoted one hour-long class to the IC module in the 8th week of the semester. During this period, the instructor demonstrated how to get started with Seattle and the first few steps of the assignment. Students then completed the module as a homework assignment.

The following year, in 2010, Professor Mache offered a networking/web-based application development course to 24 undergraduates (again mostly juniors and seniors). Again, the instructor devoted a one hour-long class (here, in the 11th week) to the IC module. Unlike the previous year, however, this time students only worked on the module during course time (in the lab). Due to other pending projects, it was not assigned as homework. Rather, the IC module was used with the hope that it would encourage some students to explore the Seattle platform in more depth for their semester project.

In both the 2009 and 2010 courses, we asked students a brief set of seven knowledge questions before they engaged in the IC module. These questions were intended to obtain a baseline of student understanding with respect to the concepts that lay at the core of the module. Our survey asked questions on three areas: symmetry/transitivity; latency; and network address translation.

We pooled responses from 2009 and 2010 since response sizes were small (7 and 18 students respectively). A good fraction of students demonstrated a priori networking knowledge by answering at least some of our knowledge questions correctly: one question on network symmetry (10 of 25), one question on network transitivity (18 of 25), three questions regarding latency (17 of 25, 16 of 25, and 16 of 25 respectively), and two questions on network address translation (16 of 25, and 3 of 25 respectively). While these results do demonstrate some clear a priori understanding of the networking properties covered by this module, there is obvious room for improvement. On average, students scored only 54% on the pre-module knowledge test. Moreover, even on the question with the highest correct score, nearly 30% of the students answered incorrectly. Overall, we feel that this pre-module knowledge assessment demonstrates that while these networking concepts are not entirely new to students, there is a need for assignments or projects to help solidify the student's knowledge and understanding.

We had intended to follow up the pre-module knowledge survey with an identical survey issued after the IC module was completed. However, due to a mix up in 2010, we only obtained post-test survey results from 2 of the students who took the pre-test. While our efforts were more fruitful in 2009 (where we obtained post-test results from all 7 of the pre-test takers), the combined results from all 9 respondents that took both pre and post tests show only a very modest improvement in test score. Average score increases slightly from 54% to 59% across the 9 respondents. We expect that we might see more robust improvements with a larger sample size and hope to be able to explore this more fully in fall 2011.

Student Perceptions

From the instructor's perspective, it seemed clear that students liked the IC module. When asked after the fact about the best part of the assignment, responses were varied, but a prevailing theme involved students actually being able to run code interactively on resources across the world. This comment typifies the theme:

"I really liked learning about distributed computing in this fashion. It was cool to see a program I initiated running on 10 foreign machines."

CONCLUSION AND THOUGHTS ON FUTURE USAGE

The Internet Connectivity module offers a very short introduction to distributed computing and helps illustrate some core networking concepts. For the instructor, setup is relatively simple as Seattle requires only python 2.5/2.6 which is standard on most Linux distributions and comes pre-installed on Mac OS 10.5/10.6.

While both of the course offerings discussed in this paper used the IC module toward the middle/end of the semester, the authors agree that the IC module is equally well suited to use at the beginning of the semester. Since the module requires no programming, it could be used before students have any significant understanding of networking to illustrate interesting (and perhaps non-obvious) features of the Internet (e.g., lack of symmetry / transitivity). Used in this manner, the IC module may provide a hands on illustration of the Internet at work that could help to motivate more discussions and projects that deal with more advanced networking and distributed systems concepts. For interested instructors, this would dovetail well with existing Seattle resources. These modules teach students about more advanced networking and help them develop real world applications.

ACKNOWLEDGEMENTS

This work is supported in part by the National Science Foundation (NSF) under award CNS-0829651 and CNS-0834243. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, GPO Technologies, Corp., or the GENI Project Office.

REFERENCES

[1] Cappos, J., Beschastnikh, I., Krishnamurthy, A., Anderson, T., Seattle: A Platform for Educational Cloud Computing, *Proc. of the 40th Technical Symposium of the ACM Special Interest Group for Computer Science Education* (SIGCSE '09), 111-115, 2009.

[2] Vegso, J., Drop in CS Bachelor's Degree Production, *Computing Research News*, 18 (2), 2006.

[3] Wallace, S. A., Bryant, R., Orr, G., The Northwest Distributed Computer Science Department, *Journal Computing Sciences in Small Colleges*, 25 (1), 143-148, 2009.

[4] Peterson, L., Anderson, T., Culler, D., Roscoe, T., A Blueprint for Introducing Disruptive Technology into the Internet, *Proc. of the First ACM Workshop on Hot Topics in Networks* (HotNets-1), *ACM SIGCOMM Computer Communication Review*, 33 (1), 59-64, 2003.

[5] Seattle: Open Peer-to-Peer Computing, 2011, seattle.cs.washington.edu, retrieved April 20, 2011.