LINKING SIMULATION AND PRACTICE

Chris McDonald
Computer Science & Software Engineering, The University of Western Australia, Crawley, Australia

Keywords: Simulation, Wireless Networks, Computer Science Education.

Abstract: The current generation of Computer Science students are far more likely to engage with computer networking through their own mobile, wireless devices than they are using fixed, wired desktop computers. Traditional approaches to teaching computer networking evolved when the Internet was composed of fixed wired infrastructure, and this historical background still forms most of the material in contemporary textbooks on computer networking. Today's students have strong expectations that their computer networking courses will have a significant focus on the networking devices and applications that they use daily - increasingly mobile and wireless. This paper describes a computer-networking project, examining *delay tolerant networks*, that has successfully assessed students' understanding of networking protocols using both simulation and actual implementations on Apple iPods. Importantly, students developed *identical* source code to be compiled and then executed by both the simulator and the iPods. The paper then reflects on the use of simulation in computer networking projects, identifying some deficiencies in students' understanding that may be masked by the use of simulation alone.

1 INTRODUCTION

It can be argued that computer networking continues to be the field of computing making the greatest impact on our daily lives, with the Internet becoming the ubiquitous carrier of global applications such as user-contributed videos and music, instant messaging, location-aware mapping and applications, and the myriad of competing and often contradictory information sources. Our students' almost insatiable desire for access to the Internet often pushes aside the desire to actually understand the technical aspects of how it all works, and even why it all works.

Contemporary students, collectively described as Generation-Y, the M-generation, and Millennials, are more likely, or wish, to access many Internet services using handheld and mobile devices, including laptop computers, network-enabled personal digital assistants (PDAs) and palmtop computers or tablets (Junco, 2007; Oblinger, 2005). For current students in Computer Science courses the Internet's "last-mile" problem is being solved, not through traditional fibre or copper connections to the home, but by near-ubiquitous wireless access. Consequently, students undertaking courses in computer networking today are expecting their courses to contain, if not focus on, details of wireless and mobile networking, developed this century, and not to remain focused on the traditional, often inaccessible, wired backbone infrastructure and their protocols, developed in the 1970s.

To maintain students' engagement and, more importantly, to increase their understanding of the design and implementation of computer networks, it is now necessary to convey many of the fundamental concepts by drawing upon mobile and wireless networking examples. This paper presents our recent successes with student projects employing a combination of simulations and real exercises using the wireless communication facilities of Apple iPods.

2 THE CNET SIMULATOR

The *cnet* network simulator has undergone a sequence of developments over eighteen years, being used as a primary teaching tool in both undergraduate and graduate computer networking courses (McDonald, 1991). Today *cnet* supports student-developed protocols and experimentation with a variety of data-link layer, network layer, and transport layer networking protocols in networks

McDonald C. (2009).
LINKING SIMULATION AND PRACTICE.
In *Proceedings of the First International Conference on Computer Supported Education*, pages 258-264
DOI: 10.5220/0001979702580264
Copyright © SciTePress

comprising any combination of wide-area-networking (WAN), local-area-networking (LAN), or wireless-local-area-networking (WLAN) links. Students working in open or closed laboratory sessions, or professors, wishing to demonstrate networking concepts in the classroom may develop or extend a wide variety of network protocols and employ *cnet* controls and statistical routines to gather and evaluate their experiments. *cnet* has been selected by William Stallings to complement the materials in his award winning networking textbook (Stallings, 2007).

cnet has most recently been extended to support wireless networking and, in particular, now supports ad-hoc and mobile networking as characterized by delay tolerant networking (see section 4). Using *cnet*'s event-driven framework students may control the mobility (speed and direction) of hundreds of mobile nodes, having them move across a physical landscape and occasionally pausing at defined landmarks or when other mobile nodes are detected. *cnet* permits full control of one or more wireless antennae on each mobile node and the physical characteristics, such as transmission frequency, output power and gain, input sensitivity, and whether the node's radio card is in a low power “sleep” mode, may all be controlled, either as fixed parameters to a single simulation or dynamically for a location- and power-aware protocol..

Students also have programmatic control over the propagation of their nodes' wireless signals. For example, once *cnet* has considered the physical properties of each transmission (to determine strength of arrival over the distance between nodes) and managed physical layer collisions, student-written code may further determine the successful arrival of a signal through obstacles (such as walls), across different terrain types (such as concrete or grass), or at certain angles (using omni-directional antennae).

cnet was first developed as a simulator running on a variety of Linux and UNIX platforms and Apple's Mac OS-X. Support has recently been added for student-written protocols to be cross-compiled for the Apple iPhone and iPod Touch platforms. The motivation of this has been so that students may first design, implement, test, and evaluate wireless network protocols within the robust simulator, and then cross-compile their *unmodified* source code for execution on the physical, mobile devices. It is believed that, in combination, the use of both simulation and actual exercises can dramatically increase a student's understanding of wireless networking, particularly the nature of physical errors and transmissions. Recent developments have been generously supported by a ACM-SIGCSE Special Project Grant and through support from Apple Inc. *cnet*'s full source-code and examples are released under the GNU Public Licence (GPL).

3 PROTOCOLS ON IPODS

The Apple iPod Touch is a handheld mobile device with a 670MHz ARM11 processor, 128MB of RAM in which the operating system and applications execute, and of flash memory providing a solid-state disk. The iPod runs a modified version of Apple's OS-X operating system and provides a multi-touch GUI for all input. Apple promotes the devices as music and video players, also capable of executing approved applications. However, it is possible to bypass the supported, constrained, environment of the device and to develop general programs for it, taking advantage of the IEEE802.11 (Wi-Fi) networking interface and general TCP/IP protocol stack (an activity requiring the unsanctioned “jailbreaking” of the device).

Our school purchased twenty 8GB iPod Touch devices for experimentation in this networking project. Our students have long developed their *cnet* simulations on their own home or laptop computers, as well as on our laboratory equipment. For this project, students were not expected to have undertaken the difficult task of installing all files necessary for cross-compilation for the iPods. Instead, the *cnet* simulator was modified to accept an additional command-line argument to request that students' protocol files be sent to a purpose-written remote *cnet*/iPod cross-compiler. The round-trip between the students' personal machines and the server was typically only three seconds. The products of this cross-compilation phase are a directory of files, forming an *application bundle*, including the actual executable application, an XML configuration, and basic icons. The salient point is that *identical* source code is developed and compiled for protocols exercised by both the simulator and the iPods.

The Students then loaded this new iPod application, their developed protocol, on (typically) six iPods and walked around one floor of our Computer Science building. Under execution, students' protocols perform as they did on the simulator, generating fictitious messages and control packets forming the implemented protocol, and responding to screen-based input via *cnet*'s event buttons or the virtual keyboard. Wireless frames from the students' protocols are actually transmitted via the iPod's Wi-Fi interface, as encapsulated UDP frames broadcast to all devices within range. A non-standard artefact of the iPod's networking protocol stack is that frames broadcast from the Wi-Fi interface are also received by the transmitting device, and *cnet*'s on-device implementation was modified to filter these frames before they reached students' protocols.

4 DTN PROTOCOLS

For the assessed project described in this paper, we required students to implement a *delay tolerant network* (DTN) protocol. A DTN is one in which mobile wireless nodes experience frequent, lengthy delays in communicating with other nodes because there is no guaranteed connectivity between communicating end points (Jain, 2004). The nodes themselves, typically personal digital assistants (PDAs), “smartphones”, or tablets, generate network traffic for the other nodes in the network, and transmit their messages using their own built-in wireless antennae. The lack of fixed infrastructure means that the nodes are collectively and cooperatively responsible for the delivery of their own messages, and for the messages of other nodes.

DTNs are often described as potential mechanisms for communication between low-earth orbiting (LEO) satellites, for communication between devices in environments lacking fixed infrastructure, and for emergency workers communicating after natural disasters. The most frequently cited example of DTNs is the Wizzy Digital Courier service which exchanges email between remote villages and schools in South Africa using motorcycles and a milk truck (Wizzy, 2007).

In an environment where wireless network coverage is sparse, such as (unfortunately) many university campuses, the mobile devices will, themselves, carry and exchange messages for each other. The devices (actually, the people carrying the devices) form a social network with other devices that they meet while walking around campus, and periodically try to keep in contact with them by generating and sending self-contained messages. The nature of the network is such that the reliable and timely delivery of the messages is not critical; thus protocols and applications in this domain must be delay tolerant. Messages eventually arriving at the intended destination may arrive out of order and do not need to be acknowledged.

There are many metrics used to evaluate the effectiveness of a DTN. For example, information in messages can lose its value over time, and if our DTN cannot deliver messages in a timely fashion, then it will be considered ineffective (and hence will not be used). The latency with which messages can be delivered will depend on a number of factors, including the number of nodes in the communication region (and thus the node density), and the speed at which nodes move (and thus come into contact with other nodes). Similarly, later messages from a source node could be considered more important than earlier ones, and devices have limited storage available for carrying the messages of other nodes. In combination, these two factors suggest that nodes may have to occasionally *drop* messages due to a lack of storage. To evaluate the effectiveness of a DTN implementation we thus need to evaluate it under a range of operating conditions.

5 THE DTN PROJECT

The *Computer Networks* course at The University of Western Australia is offered to students in their third undergraduate year, and is taken by a wide variety of students undertaking bachelor degrees in Computer Science, Software Engineering, and Electronic Engineering. In recent years the course has had enrolments of between 120 and 65 students. The course comprises one eighth of a year's work, and is assessed with a practical project contributing 40% and two examinations contributing 60%. The practical project, described in this section, ran for 6 weeks and was undertaken in groups of four.

The goal of the student project was to design, implement, and analyse a delay tolerant protocol using the *cnet* simulator, to analyse and report on the observed effectiveness of the protocol developed, and to report on the differences between the simulated protocol and the actual, physical one. The project was designed to assess students' understanding of important aspects of Media Access Control (MAC), Data Link and Network Layer networking protocols, and their ability to analyse and report on their observations.

In some schools, such a project may appear under-specified, as no explicit information was provided as to how students should undertake their protocol development and evaluation. Students had already completed four weekly closed laboratory sessions, which had introduced them to the *cnet* environment and, in particular, encouraged them to experiment with the fundamentals of wireless signal transmission and reception, typical transmission distances, and the handling of frame collisions.

The research topic of delay tolerant networks is sometimes described as a “solution without a problem” and, as an artefact of this, there are extremely few general-purpose protocol implementations available for consideration or even described in textbooks. For this reason, we view delay tolerant networks as a fertile ground for student projects in which students are expected to demonstrate initiative in developing and analysing their own designs. Projects involving delay tolerant networks hold many opportunities for creativity, as there are so many protocol characteristics that can be varied. Another beneficial consequence was that no internet-sourced plagiarism was observed.

6 STUDENT EXPERIENCES

Students undertook the project in 17 groups of four, but often chose to co-opt other students to carry additional iPods to form DTN networks of up to 15 nodes. About ten students also used their own iPhones and iPods in the groups' experiments. Student groups typically partitioned the networking responsibilities into lower and upper layers. Prompted by lecture materials, one pair of students developed a carrier-sense multiple-access with collision-avoidance (CSMA/CA) MAC protocol to transmit wireless data frames, and a simple mobility model (both only required on the simulator). The other pair developed the node beaconing and message transmission and carrying responsibilities. *cnet* protocols are written in ISO-C99, and the C-preprocessor is thus employed to isolate the simulator-specific code.

Students typically implemented a simple *random waypoint* model of node mobility, in which nodes would randomly choose their next remote destination and walking speed, walk toward that destination at a constant speed and, once there, pause for a random period of time before setting off again (Camp, 2002). At first, students anticipated that they would experience difficulty with both walking and transmitting and receiving simultaneously (generation-Y's equivalent of being able to walk and chew gum?), but *cnet*'s internal scheduler represents and reports events, such as periodic movement and wireless frame arrival, consistently, making the anticipated complexity disappear.

To provide a strong sense of localization and realism, our students were provided with a representation of the floor plan of our Computer Science building as *cnet* enables this map to be displayed on the simulation background. The random waypoint model was modified to honour the position of walls in our building, and moving nodes could only enter corridors and lab areas through the actual doorways. Similarly, *cnet* protocols (under simulation) have the opportunity to define their own wireless signal propagation models. Earlier experiments had identified that the Wi-Fi signals of the iPods could pass through one, but not two, of the (concrete block) walls in our building, and students implemented a signal propagation model that dropped signals attempting to pass through two or more walls. A representative student DTN simulation is presented in Figure 1.

As well as developing solutions for mobility and wireless signal propagation, students had to design and implement a strategy for message generation and delivery. The simplest metric for evaluating the success of a strategy was, while holding constant the number of nodes and the walking speeds and pause durations, to measure the average delivery time of each message (*cnet* permits all simulation attributes and required statistics to be specified in an external topology file). During the project it was very encouraging to see students using our asynchronous message forum to independently promote a competition to find the shortest possible message delivery time.

From the students' submissions it was clear that most groups investigated a number of strategies of increasing complexity, driven by the desire to minimize message delivery time. The simplest strategy seen amongst the students' submissions was one making no attempt to transmit a generated message until the intended recipient was within wireless range (such nodes periodically transmitted beacon frames). While this strategy works, messages would typically take several *minutes* to be exchanged between nodes that appeared within the same region by chance (although the presence of social hotspots increased the likelihood of their meeting). The next strategy attempted was one in which nodes quickly transmitted their messages to any close neighbour, attempting to simply “pass-off” the responsibility for delivery (akin to the basic “hot potato” routing algorithm often cited in networking textbooks). Students quickly observed that this strategy frequently resulted in two adjacent nodes continually exchanging messages until one of them commenced walking. The solution to this required nodes to either mark messages that they had most recently held, or for nodes to maintain queues of recently seen messages (to thwart cyclic exchanges).

The most complex student strategies attempted involved nodes not only maintaining queues of outgoing and recently seen messages, but also maintaining information about recently seen nodes and, using the signal strength of frames arriving from those nodes, estimates of the direction and speeds of those nodes to determine if they would still be within wireless range. These advanced solutions involved nodes “bidding” to forward an offered message in a direction in which the intended recipient was seen very recently. In later simulations involving as many as three hundred nodes on our campus (i.e. high node densities in an environment in which signals could propagate 70 metres), message delivery times were often less than a second.

7 REFLECTING ON SIMULATION

As recently as 2002, large enrolments in the discipline of Computer Science resulted in a number of modifications to our teaching methods to address the problems associated with teaching large cohorts of students. Changes included the dropping of weekly tutorials, an increased use of software simulations to demonstrate many Computer Science concepts, and the development and support of flexible learning approaches facilitating use of home and laptop computers.

More recently, the dramatic drop in enrolments in Computer Science has not been matched by a return of some of our traditional teaching methods, such as tutorials and in-class and in-laboratory demonstrations and experiments. Students are still exposed to high quality teaching materials and exercises, that are typically undertaken alone, but the student experience is considerably poorer for the reduced interaction with other students, and a reduced focus on experimental techniques involving physical, error-prone, scenarios. Related concerns are also raised by Kotz, who identifies a number of simplistic assumptions made by many wireless network simulation tools (Kotz, 2004). Excessive use of simplistic simulations, without accompanying physical exercises to both support and verify the observations made from the simulations, should raise the obvious concerns about what is actually being taught to, and learnt by, our students.

The dramatic reduction in Computer Science enrolments is not, of course, specific to our university, and is being widely experienced in the western hemisphere. Prospective Computer Science students, and even those already within our programmes, have realistic concerns about the “offshoring” of programming skills, and the relevance of some Computer Science material to their future careers. It is not unreasonable for students to question the value of their university contact, if they can ``successfully'' undertake all learning online, and in isolation.

A wealth of recent literature in Computer Science Education journals and conferences focuses on teaching practices to revitalize the discipline, and to increase the on-campus engagement of students. In particular, many universities are reintroducing practical, experimental-based laboratory sessions in which students see an immediate benefit to undertaking the work at university, and with other students.

Our simulation-based investigation into delay tolerant networks, which was successful in achieving its initial aims, has highlighted a number of problems that must be addressed in the near future. The type of project described in this paper is well supported by the *cnet* simulator, making it widely available to a large number of universities and colleges. However, while students were able to successfully demonstrate their ability to develop reasonably complex network protocols, they experienced genuine difficulties developing hypotheses, constructing and executing (simulation-based) experiments, and evaluating the observed results. Furthermore, it was clear that students had great difficulty relating the physical dimensions and properties, represented in their simulation, to an actual wireless computing environment that they observed with the iPods, and it is believed that this is affecting their ability to construct meaningful experiments that will expose and demonstrate the strengths and weaknesses of their protocols.

To address the observed shortcomings with students' understanding we are currently developing a sequence of additional projects that will combine the convenience and versatility of network simulation, with the error-prone physical characteristics of real wireless networking. New, smaller, projects will incorporate actual mobile wireless devices into our course. The motivation behind this is the belief, supported through our observations with this reported project, that many Computer Science students currently have a poor understanding of the physical constraints of mobile and wireless networking, and that this is limiting their understanding of how, and why, contemporary wireless protocols have to address these constraints. In addition, it is believed that knowledge of this material is not currently developed when only exposing students to simulated network environments.

8 SUMMARY

This paper has described a novel student project in computer networking focusing on the types of networking with which contemporary Computer Science students are most familiar - mobile and wireless networking. Within the project, students are still exposed to many of the traditional aspects of networking including the need for error detection and recovery, the value of layered network protocols, and the challenges of robust and reliable message delivery. The project encourages students to design and implement a sequence of solutions of increasing complexity, and our experience has been that students have been enthused by the challenges that this project has raised.

Interested readers are welcome to contact the author for copies of the full project materials described in this paper, including the cnet network simulator and its support for iPods.

REFERENCES

Camp, T., Boleng, J., Davies, V., 2002. A Survey of Mobility Models for Ad Hoc Network Research, In *Wireless Communications & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, v2, no 5, pp483-502.

Jain, S., Fall, K., Patra, R., 2004. Routing in a delay tolerant network, In *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, Portland, Oregon, pp145-158.

Junco, J., Mastrodicasa, J., 2007. Connecting to the Net.Generation: What Higher Education Professionals Need to Know About Today's Students, In *Student Affairs Administration in Higher Education (NASPA)*, 164pp.

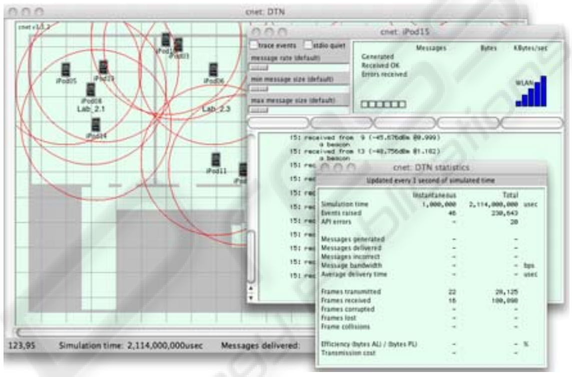
Kotz, D., Newport, C., Gray, R.S., Lui, J., Yuan, Y., Elliot, C, 2004. Experimental evaluation of wireless simulation assumptions. In *Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pp78-82. ACM Press.

McDonald, C.S., 1991. A Network Specification Language and Execution Environment for Undergraduate Teaching, in *Proceedings of the ACM Computer Science Education Technical Symposium '91*, San Antonio, Texas, pp25-34. Software available from www.csse.uwa.edu.au/cnet3/.

Oblinger, D.G, Oblinger, J.L., Editors, 2005. Educating the Net Generation, EDUCAUSE, available from www.educause.edu/books/educatingthenetgen/5989,.

Stallings, W., 2007. *Data and Computer Communications*. 8th ed., Prentice-Hall.

The Wizzy Digital Courier, 2007. Available from www.wizzy.org.za.



```
15: received from  9 (-45.676dBm @0.999)
        a beacon
15: received from 13 (-48.756dBm @1.182)
        a beacon
```

```
                                  Instantaneous            Total
Simulation time                       1,000,000    2,114,000,000  usec
Events raised                                46          230,643
API errors                                    -               20

Messages generated                            -                -
Messages delivered                            -                -
Messages incorrect                            -                -
Message bandwidth                             -                -  bps
Average delivery time                         -                -  usec

Frames transmitted                           22           20,125
Frames received                              16          100,000
Frames corrupted                              -                -
Frames lost                                   -                -
Frame collisions                              -                -

Efficiency (bytes AL) / (bytes PL)            -                -  %
Transmission cost                             -                -
```

Figure 1: A representative student DTN simulation.