VERKKOKE: Learning Routing and Network Programming Online *

Anton Alstes
Helsinki University of Technology
P.O. Box 5400
FIN-02015 TKK, Finland
aalstes@tml.hut.fi

Janne Lindqvist
Helsinki University of Technology
P.O. Box 5400
FIN-02015 TKK, Finland
janne.lindqvist@tml.hut.fi

ABSTRACT

We present an Online Teaching Environment (OTE) that supports "learning by doing" philosophy in teaching telecommunications software and routing. "Learning by doing" is achieved by giving students a programming assignment that introduces them to socket programming and gives them the opportunity to practice implementing simplified routing protocols. The OTE creates individual assignments for students, accepts solution submissions via the Internet, and, finally, checks the assignments automatically. The system also notifies the students of possible mistakes in their solutions, so they can learn from their mistakes, fix them and resubmit the corrected solutions. The teacher needs only to start the system when the course begins and verify the assignment results when students have finished their work.

The OTE is compatible with modern learning management systems through its adherence to the Sharable Content Object Reference Model (SCORM) specification. The OTE supports intricate and realistic programming assignments through representative topology generation and sophisticated network simulation.

Categories and Subject Descriptors

K.3.1 [**Computers and Education**]: Computer Uses in Education—*computer-managed instruction (CMI), distance learning*

General Terms

Algorithms, Human Factors, Theory

Keywords

Learning environment, programming, routing

*The online learning environment described in this paper has previously been introduced as a one-page abstract [9]. The implementation can be downloaded from http://www.tml.tkk.fi/Research/VERKKOKE/

ITiCSE'07, June 23–27, 2007, Dundee, Scotland, United Kingdom.

1. INTRODUCTION

As the importance of the Internet continues to increase, telecommunications software and routing become ever more essential topics for computer science students to learn. Understanding how the Internet and telecommunications networks in general function is imperative for the computer scientist or engineer in a world where interconnected computers enable applications that play such a significant part in our everyday lives: the World Wide Web, electronic mail, instant messaging, Internet telephony, video conferencing, to name only a few.

Various teaching methods can be used to educate our future computer scientists and engineers in the areas of telecommunications software and routing. The traditional method has been to use lectures, hand-outs, books and written assignments as teaching aids. These traditional teaching aids will probably continue to be useful also in the future, but the process of learning telecommunications software and routing can certainly benefit from more innovative approaches.

To support "learning by doing" teaching philosophy on the Computer Networks course in Helsinki University of Technology, a requirement for passing the course is a programming assignment. The programming assignment introduces the student to socket programming and gives the student the possibility to practice implementing simplified routing protocols.

The solutions to the programming assignments have previously been demonstrated to assistants and the reports were submitted as paper printouts and all students have done the same assignment. The students have complained that the submission method is out-of-date and inappropriate for a course about computer networks since the course curriculum includes many examples of applications that could be used to submit the assignments over the Internet.

To answer to the above requirements and to reduce the work load of the course personnel without hindering the students' learning process, we have implemented an Online Teaching Environment (OTE) for the computer networking course.

The OTE creates individual assignments for students, accepts solution submissions via the Internet, and, finally, checks them automatically. The system also notifies the students of possible mistakes in their solutions. This way, the students can learn from their mistakes and fix them and resubmit the corrected solutions. The goal of the implementation is that the teacher only needs to start the system when the course begins and verify the assignment results when students have finished their work.

The rest of the paper is organized as follows: the next section discusses related work, that is, other software for teaching networking and/or programming. Then, we describe the implementation of the OTE, followed by discussion, that is, we asses its suitability for teaching telecommunications software and routing and comparare it with other systems. Finally, we conclude the paper and discuss work to be done in the future.

2. RELATED WORK

2.1 The Virtual Network System

The Virtual Network System (VNS) [4] is a teaching tool developed at Stanford University. Like the OTE, it was designed to support programming assignments of an undergraduate introductory networking course.

Like the OTE, VNS can simulate multiple network topologies and provides a server that accepts connections from clients. VNS logically makes the clients operate as hosts on the simulated network topologies. VNS achieves this by using a client side component which is a set of libraries, that the student's client code must use. The libraries handle interacting with the VNS server, over TCP. They provide to the student's code the raw Ethernet traffic received by the simulated virtual host, as well as allow the student's code to inject packets back into the network.

According to Casado and McKeown [4], VNS can be used in any project requiring low-level network access, e.g. in teaching the implementation of a router. Because VNS makes students see the raw Ethernet layer, students' implementations must support low-level functionality such as address resolution or checksum calculation.

2.2 J-Sim

J-Sim [16] is a simulation environment written in Java. It is based on an autonomous component architecture (ACA), similar in its design to integrated circuits (IC). The analogy is between hardware modules composed of IC chips and software systems built from ACA components.

J-Sim includes a generalized model of packet switched network, built on top of the ACA that can be used for network modeling and simulation. Using the components of the network model, one can create simulated networks consisting of nodes (end hosts or routers) and links. J-Sim provides components for defining the inner structure of nodes, such as their network interfaces, routing table, routing protocols, transport protocols and applications.

J-Sim includes a Tool Command Language (TCL) scripting environment that allows setting up simulation scenarios with the desired topologies, node structures and network protocols. Thus it could be used as a tool for teaching network protocols such as dynamic routing protocols. Teaching network programming, on the other hand, with J-Sim alone, is more difficult, because J-Sim does not provide a server for students' clients to connect to.

2.3 Ludwig

Ludwig [15] is a web-based programming tutoring and assessment system, developed at Penn State University. It provides a controlled text editor, implemented as a Java applet. Students write their programs using the editor, which does not allow pasting text from other sources. When the first version of the program is ready, the student submits it for analysis. The system first attempts to link and compile the program. If this fails, the failure messages are shown to the student, who makes the required modifications to the program and resubmits. Upon successful compilation, Ludwig does a style check to the program. Stylistic items that are checked include proper indentation and cyclometric complexity.

Ludwig then runs the program with input data prepared by an instructor or by the student. The same input is given to a solution program developed by the instructor and the outputs of the programs are compared. Differences are shown to the student, who can either submit the program for grading or modify it to get a better result. When the student submits the program for grading, a different data set is used as input, to prevent students from simply submitting a program that prints the expected output without actually processing the input.

Although Ludwig was designed for teaching introductory programming, it could be conceivable to use it for teaching network programming as well.

2.4 TRAKLA2

TRAKLA2 [8] is a framework for building interactive algorithm simulation exercises. The exercises are based on Java applets that visualize data structures. Students are expected to simulate the operation of algorithms by using drag-and-drop operations to manipulate the visual algorithm representations. The data structures should be changed by the students as the algorithm in question would do. The resulting sequence of data structure states form the student's answer to the given problem. TRAKLA2 automatically grades the student's solution and provides a model solution, which is an animation that visualizes the operation of the algorithm step by step.

Since TRAKLA2 is designed for teaching algorithms and data structures in general, it is a useful aid in teaching algorithms used in routing protocols, e.g. Dijkstra's algorithm.

3. ONLINE TEACHING ENVIRONMENT

The OTE supports the programming assignment of our introductory networking course by providing a socket interface that clients programmed by students can connect to. To allow students to practice implementing simplified routing protocols, the server emulates a network consisting of several routers. A student's client represents one router whose routing table is calculated on the basis of the information sent by the server.

The OTE creates individual assignments for students, accepts solution submissions via the Internet, and checks them automatically. The system also notifies the students of possible mistakes in their solutions. This way, the students can learn from their mistakes, fix the mistakes and resubmit the corrected solutions. The teacher only needs to start the system when the course begins and verify the assignment results when students have finished their work.

More specifically, the OTE has the following capabilities:

- It generates network graphs that are used with the routing algorithms.
- It provides a management interface for enrolled student management. This web-based interface also produced reports on student assignment statuses.

- It provides a standard network socket interface that acts as a server to client programs developed by the students.
- It implements abstractions of two routing protocols: one of the distance vector family and one of the link state family. Abstractions of real routing protocols (such as Routing Information Protocol (RIP) [7] or Open Shortest Path First (OSPF) [11]) are used to communicate the basic ideas without overwhelming students with unnecessary details.
- It is able to compute routing tables and check whether student submitted routing tables are correct.

3.1 OTE Components

The OTE consists of the following main parts:

- A simulation server that simulates a network and runs routing protocols within that network. The simulation server also accepts socket connections from clients programmed by students and forwards routing messages to those clients.
- A Sharable Content Object Reference Model (SCORM) compliant Learning Management System (LMS), e.g. Moodle [1]. The student starts the assignment from within the LMS and the assignment results are shown to the students (and teachers) in the LMS. The Web user interface of the OTE is packaged into a SCORM-compliant Sharable Content Object (SCO), except for the dynamic server-generated part, which resides in a frame of the SCO.
- A server-side component that connects the LMS and simulation server together. This component is invoked from the LMS when the student starts the assignment. It generates a topology for the student that will be simulated in the simulation server. It also gets the assignment result from the simulation server and gives it to the LMS.

We built the simulation server by incorporating J-Sim as the simulation engine and extending it with a component that listens to protocol messages of a specific simulated network node and forwards those messages to the student's client. We also implemented a socket server to interface with the client. Effectively, the student's client corresponds to a simulation node, i.e. the client receives via the socket the protocol messages that the simulation node receives. The component that listens to the protocol messages only forwards the messages of interest, i.e. routing protocol messages.

To provide students with individual simulation scenarios, the OTE uses the Brite topology generator to produce random representative network topologies for each student. Brite is described in a paper by Medina et al. [10].

To provide students with a visual representation of the generated topology, the OTE uses the Graphviz open source graph visualization software. Graphviz is described in a paper by Gansner and North [6].

An LMS is used by the OTE for managing student assignment statuses and results, submitting documents related to the assignment, e.g. assignment plans, for reporting purposes and for other value-adding functionality that a particular LMS may provide. According to an article by Paulsen [12] that presents findings from analyses of the European Web-edu project, many universities have purchased an LMS and LMSs seem to work satisfactorily in various educational institutions throughout Europe. Since a university giving a networking course is likely to already have an LMS in use, it is usually desired to use this existing system to manage course information, such as students and their assignment scores.

SCORM is a standard of the Advanced Distributed Learning (ADL) initiative of the United States government. According to Bohl et al. [3], SCORM has much potential, but it is not without restrictions. Despite the restrictions, we decided to comply with the SCORM standard because it is widely implemented in available LMSs, which allows portability and reusability, and because SCORM provides a means to exchange important metadata between the content and the LMS, such as assignment states and scores.

3.2 Using OTE

To do the programming assignment, the student takes the following steps (see Figure 1):

- Step 1: The student logs in to the LMS to read assignment instructions, to commence the assignment and retrieve assignment results.
- Step 2: Externally, the student makes the client program, then logs back into the LMS to start the assignment.
- Step 3: The system generates a random topology for the student.
- Step 4: The student uses the client program contact the simulation server. The simulation server sends routing messages to the client based on the topology generated for the student.
- Step 5: The client builds its routing table based on the routing messages and sends the routing table to the server. The simulation server checks whether the routing table is correct and stores the result in the LMS. The student goes back to step 1, to see the result. If the routing table is not correct, the student can optionally be allowed multiple resubmits.

The student's client builds its routing table based on routing messages sent by the server. The format and content of these messages depends on the simulated routing protocol. In its initial version, the OTE simulates two routing protocols which are abstractions of real routing protocols. The abstractions are called DV (which stands for Distance Vector) and LS (which stands for Link State), and they are loosely based on the real protocols RIP and OSPF, respectively.

The DV protocol uses the User Datagram Protocol (UDP) [13] so that the student gets experience with datagram sockets. As the student's client contacts the server, it sends an authentication message that identifies the student. Then the server starts sending the routing data. The routing data messages have the format "[x, y, z]", where x is the network interface number, y is the name of a router, and z is the distance to router y. Each of these messages are sent in separate UDP packets. In the DV protocol, the distance between two adjacent routers is always assumed to be one. As the client

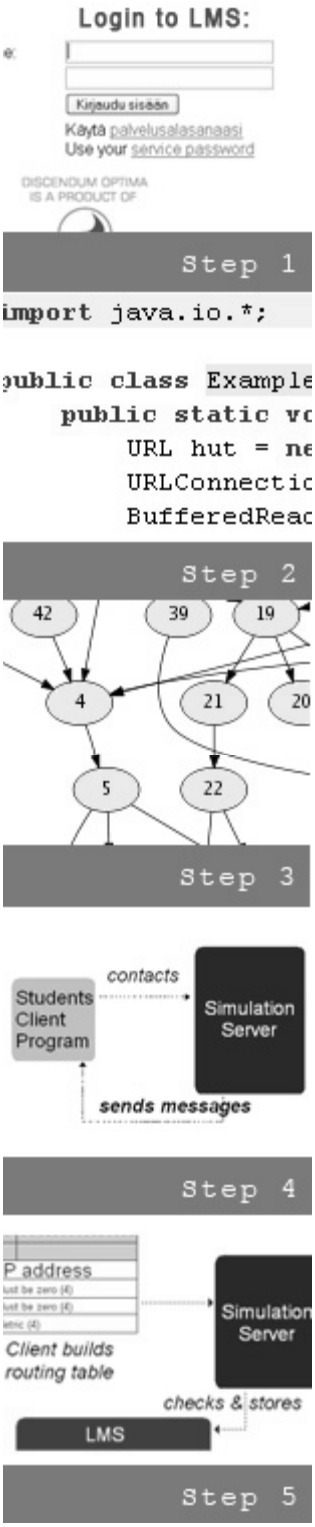

Figure 1: Programming assignment steps. See description in text.

receives the routing messages, it calculates its routing table using a distributed Bellman-Ford algorithm (based on the original non-distributed Bellman-Ford algorithm [2]).

The LS protocol uses the Transmission Control Protocol (TCP) [14] so that the student gets experience with byte stream sockets. Like with the DV protocol, the student's client first sends an authentication message that identifies the student. Upon successful authentication, the client sends a *HELLO* message to the server. The server responds with a message for each neighbor of the student's router in the simulated network. The messages contain the names of these neighbor routers. The communication then continues according to the protocol, and finally the student's client will have enough information to build a link state database and calculate its routing table using Dijkstra's shortest path algorithm [5].

The student's client sends its routing table to the server using a separate submission protocol which uses TCP. The server then compares the submitted routing table with the routing table of the student's node in the simulation.

4. DISCUSSION

The OTE supports a "learning by doing" approach to teaching telecommunications software and routing. It does this by allowing students to write their own client programs, and notifying the students of possible mistakes in their solutions, so that the students can fix them and resubmit the corrected solutions. Automatic correction gives fast feedback and allows students to learn from their own mistakes because they can still remember what they have done. We believe this will have a strong positive impact on learning results.

Since the submission is done using the Internet, students are freed from the constraints of time and place since course assignments can be solved when and where students want to do them.

VNS, discussed in the *Related Work* section, also supports the "learning by doing" approach. VNS is, however, better suited for teaching low-level networking because it forwards the students raw Ethernet traffic. Thus, it is not very suitable for teaching dynamic routing protocols on an introductory networking course, because of the amount of programming work needed to accomplish even a simple router that handles static routing, given the low-level issues that need to be addressed. VNS does also not provide assignment submission or correction functionality, meaning that course personnel must use their resources to correct the student assignments.

The OTE, on the other hand, can well be used to teach dynamic routing even on an introductory networking course, because students get the simplified routing data in a simple format over UDP or TCP, using e.g. Java sockets. Thus, the higher level operation and logic of dynamic routing can be communicated, without overwhelming students with too much detail.

The OTE's automatic correction of assignments gives fast feedback to students and reduces the work load of the course personnel.

In contrast with Ludwig, discussed in the *Related Work* section, the OTE does not, at least in the initial version, perform any style checks to the code submitted by students. Because programming style is a very personal matter, and since students can choose from more than one programming lan-

guage when implementing the client, automatic style checks would be hard to implement. Ludwig also attempts to compile the submitted program, which is needed for the operation of the system, but in the OTE such functionality would be of little value to students, since students can compile their programs with their own compilers. The other educational aspects of using Ludwig or the OTE are similar, apart from the fact that the OTE teaches routing and network programming, not just introductory programming.

5. CONCLUSIONS AND FUTURE WORK

We believe that our system will have a strong positive impact on learning results. Routing is one of the most difficult concepts related to computer networks. Theoretical studying alone is not sufficient for understanding the real practical problems. Our system allows combining theory and practicalities since theory presented in the course lectures can be immediately applied in students' own work in a very concrete way. Additionally, the students gain practice in writing real network programs that need to authenticate across the Internet and implement algorithms based on data received from the network. Future enhancements to the OTE include adding support for teaching other network protocols in addition to the dynamic routing protocols. We believe that our platform is generic enough to be extensible to all kind of networking assignments. The system has been deployed and used first time during the Autumn 2006 course and based on the feedback, the students felt that the new assignment structure and the teaching environment was good and helped the learning process. However, further studies are needed to understand how the students perceive the system and how it actually helps learning.

6. ACKNOWLEDGMENTS

The authors thank Sara Vuori for providing the assignment step figures and for other substantial assistance in the project. Also, the authors thank Sanna Suoranta for her comments and for letting the authors to mess up her course's programming assignment.

7. REFERENCES

[1] Anon. Moodle website. URI: http://moodle.org/.

[2] R. Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16(1):87–90, 1958.

[3] O. Bohl, J. Schellhase, R. Sengler, and U. Winand. The Sharable Content Object Reference Model (SCORM)A Critical Review. In *ICCE '02: Proceedings of the International Conference on Computers in Education*, page 950, Washington, DC, USA, 2002. IEEE Computer Society.

[4] M. Casado and N. McKeown. The virtual network system. In *SIGCSE '05: Proceedings of the 36th SIGCSE technical symposium on Computer science education*, pages 76–80, New York, NY, USA, 2005. ACM Press.

[5] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, December 1959.

[6] E. R. Gansner and S. C. North. An open graph visualization system and its applications to software engineering. *Software – Practice and Experience*, 30(11):1203–1233, 2000.

[7] C. L. Hedrick. RFC 1058: Routing Information Protocol, 1988.

[8] A. Korhonen, L. Malmi, and P. Silvasti. TRAKLA2: a framework for automatically assessed visual algorithm simulation exercises. In *Proceedings of Kolin Kolistelut / Koli Calling – Third Annual Baltic Conference on Computer Science Education*, pages 48–56, 2003.

[9] J. Lindqvist and S. Liimatainen. VERKKOKE: online teaching environment for telecommunications software and routing. *SIGCSE Bull.*, 38(3):319–319, 2006.

[10] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: An Approach to Universal Topology Generation. In *MASCOTS '01: Proceedings of the Ninth International Symposium in Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'01)*, page 346, Washington, DC, USA, 2001. IEEE Computer Society.

[11] J. Moy. RFC 2328: OSPF Version 2, 1998.

[12] M. F. Paulsen. Experiences with Learning Management Systems in 113 European Institutions. *Educational Technology & Society*, 6(4):134–148, 2003.

[13] J. Postel. RFC 768: User Datagram Protocol, 1980.

[14] J. Postel. RFC 793: Transmission Control Protocol, 1981.

[15] S. C. Shaffer. Ludwig: an online programming tutoring and assessment system. *SIGCSE Bull.*, 37(2):56–60, 2005.

[16] H.-Y. Tyan, A. Sobeih, and J. C. Hou. Towards Composable and Extensible Network Simulation. In *19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)*, 2005.