Migrating a Traditional Network and Data Communication Laboratory Course to an Information Systems-Friendly Environment

Michael E. Battig – mbattig@smcvt.edu
Computer Science Department, St. Michael's College
Colchester, VT 05439

and

Ronald Sobol – Ronald_Sobol@res.raytheon.com
Raytheon Electronics Systems – MS T3MR5, 50 Apple Hill Drive
Tewksbury, MA 01876

Abstract

Networking and data communication have become more prominent in the information technology arena over the past ten years. Graduates of Information Systems or Computer Science programs should possess some competence in this computing sub-discipline. However, many universities opt to exploit resources and find synergy between the Computer Science and Information Systems curricula where possible. We present an approach to teaching the subject that gives students a rich set of laboratory experiments and yet is appropriate for both the Information Systems and Computer Science curricula. Our approach gives students access to the implementation detail of data communication protocols in an NT/Visual Basic programming environment that is friendly to Information Systems.

Keywords: Networking, telecommunications, laboratory course, computing curricula, information systems curricula

1. INTRODUCTION

With the explosive growth of computer networks and telecommunications in recent years, the need for undergraduate instruction in the subject has become apparent. This particular computing sub-discipline represents one with significant overlap and synergy between IS and CS. CS curricular guidelines place this topic under the Operating Systems umbrella (ACM 1990), whereas IS curricular recommendations tend to make it a separate category altogether (AITP 1997). This paper presents a course designed to serve both the CS and IS audiences.

The curricular guidelines for CS list the following topics as part of a networks course: architecture, protocols, switching, routing, LANs, data security, and layers of communication networks. The curricular guidelines for IS list these topics: networking requirements, hardware, software, architectures, topologies, protocols, and security. Although the above lists are incomplete (we've emphasized the similarities), they do highlight the points of intersection between the curricular recommendations. Furthermore, we will show how our laboratory experiments reinforce these topics.

Many universities have the luxury of maintaining distinct IS and CS faculty/curricula. Some colleges, whether due to budget, size or curricular choice, have decided to integrate some of the IS and CS curricula. Therefore, adapting courses to both programs is preferred for the same reasons that class libraries are preferred in software development, i.e., we get greater reuse from our efforts. Our program features a joint first-year curriculum for IS and CS majors. This model has proved beneficial in a context where the driving force was service to student needs, as opposed to just optimizing faculty resources (Harrington 1995). This paper will describe experiences in developing and adapting an undergraduate course in networks/data communications and provide resources for implementing the course within any IS program.

This course began at a large state university that received an NSF grant to develop an inexpensive data communications laboratory for use in its undergraduate CS program (Smith 1994, 1991). The results produced a highly portable course concept. As part of the grant, hardware specifications for a simple data communications lab were given which involves connecting PCs together with null modems using their serial ports. The course was subsequently adapted to fit a small college curriculum and to incorporate exposure to object-oriented software development (Battig 1998). In this work we will share our experience from migrating this laboratory course to serve a joint curriculum in CS/IS. The course was adapted to allow the laboratory component to operate on Windows NT workstations using Visual Basic as the programming language to implement a simple data communications protocol. The protocol utilized is a simplification of BSC (IBM's Binary Synchronous Communications). Details of this protocol can be found in most popular textbooks (Shay 1999; Stallings 2000; Tannenbaum 1996).

2. IS/CS CURRICULAR ISSUES

Our institution offers both CS and IS degrees at the baccalaureate level. Since we are operating at a relatively small college, the CS and IS programs share faculty, computing resources, and contain a common core of courses. The goal is to create a network/data communications course that is appropriate for both majors. It is worth mentioning at this point that the IS program has an emphasis on software development and therefore leans more toward the technical (versus managerial) end of the spectrum.

Regardless of the degree program (i.e., CS versus IS), the main pedagogical goal of the networks/data communications course is to provide students with a foundation in the terminology, concepts, and implementation issues of this computing sub-discipline. The inclusion of the laboratory component is designed to reinforce the classroom material. Thus, students become intimately acquainted with at least one protocol and its accompanying frame formats. As a result, students develop a point of reference for understanding other protocols and their underlying implementation details.

There have been a number of interesting side effects observed with the inclusion of this course in the curriculum. Many students have commented that they learned quite a bit of software engineering practice from the assignments in the course. This is due to several facets of the laboratory work. First, it is an opportunity to move away from the "programming in the small" and toward "programming in the large." Since the assignments constitute several pieces of one larger puzzle, the students must live with the consequences of earlier design decisions (They are warned about this by the way, lest the reader think the authors enjoy some twisted form of hazing!). Second, this pedagogical approach is being promoted on a national level by such groups as the ACM that urge us to teach software engineering in the warp and woof of the curriculum, not just in a single course (Johnson 1997).

The nature of network layers (i.e., OSI model) lends itself well to incremental and object-oriented programming. The students begin to see the benefits of code reuse in that they are not building the entire project from scratch. Thus, the inclusion of the data communications/networks course provides an opportunity for additional exposure to the object-oriented software development paradigm in a realistic scenario.

3. LABORATORY MIGRATION

Just as abstract data types become meaningful to students through implementation, communication protocols are better understood when actually implemented. Although the possibilities are numerous, we restricted our course to four lab assignments: *send, receive, error-checking*, and *data encryption*. The assignments are given to correspond with the timing of the relevant material covered in lecture (e.g., the error-checking assignment is given at about the same time that the CRC method is discussed in class). Thus, the laboratory work consists of incrementally building a complete program that contains portions of a typical data communications exchange occurring at the lowest three levels of the OSI model.

The *receive* assignment is given first. Along with explanation of the BSC protocol, students receive instruction on how to use the hardware and software in the data communications laboratory. For the receiver lab, students are provided with a *send* program which allows for adequate testing of the *receive* program using a monitor program. For the second assignment students are required to add sending capability. Thus, the program must be able to function as both sender and receiver depending on the situation. The third assignment consists of implementing a 16-bit CRC error detection algorithm. Finally, students implement a data encryption algorithm (students are permitted to choose among several alternatives). Consequently, they are incrementally building a single program over the course of the semester.

The Monitor program, along with the others described here, is provided in the Appendix. The purpose of the Monitor is twofold. First, it provides students with the necessary means of testing and debugging their lab work. Second, it provides the instructor with the capability of verifying student work. Related to this, the monitor has the ability to corrupt CRC values in data frames. Thus, students and instructors can simulate the anomalous occurrence of data loss or corruption due to such hard to simulate events as electromagnetic interference.

Originally the lab portion of this course was migrated from Turbo Pascal (since the original NSF funded program provided Turbo Pascal source code drivers for the IBM PC communications ports) to C++ (Battig 1998). The transition from Pascal to C++ was expedited by incorporating C++ code provided by one author's RS-232 class (Nelson 1992). However, the prior lab software utilized DOS-specific routines to program the communication ports. Therefore, since we were creating new lab software (in light of our desire to use NT), we decided that the use of Visual Basic would provide a CS/IS curricular integration benefit.

Our initial experience using the new platform was positive. During the first semester of use (Spring 2000) we observed several benefits. First, Visual Basic is well suited for the interface portion of the project. I.e., students found it much easier to create separate windows for the various lab components, which allowed them to focus more effort on the essential issue of dealing with the communication protocol. Second, we found that the Microsoft Winsock 5.0 control (available to both Visual Basic 5.0 & 6.0) provides a convenient means of establishing connections between stations in the NT environment. The Winsock control provides a protocol property that allows for the specification of either the TCP or the UDP protocol. The programs provided in the Appendix demonstrate the use of this feature. Lastly, we noted that student enthusiasm increased in the new environment (we were no longer asked: "Why do we have to do our labs on this arcane equipment?").

Over the years we have found that implementing the CRC error detection algorithm presents the greatest challenge to students. In reality, we are asking them to implement something that is really suited for a hardware implementation (shift registers and XOR gates are not typically implemented in Visual Basic). Therefore, we are providing a complete CRC test program (written in Visual Basic, see Appendix), which may be supplied to students as part of laboratory materials for the course. Thus, the students will need to incorporate the CRC routine into their communication program in order to validate the integrity of the data during the transmission (in reality the monitor poses the greatest statistical threat to their data given the likelihood of data corruption on a typical campus network).
We have discovered that the use of Visual Basic produces some interesting outcomes. First, most IS majors have some background in VB. Second, CS majors (who are typically familiar with C++ and/or Java) are able to learn Visual Basic quickly. This additional programming language exposure has some positive side effects. Students implicitly learn programming language differences (especially given the *event driven* nature of VB). Students have also reported that Visual Basic programs are fairly robust (compared to the DOS/C++ environment) and are easier to test and debug.

4. CONCLUSION

The inclusion of a networks and data communications course to the synergistic IS/CS curricula has been presented. We have shown that points of intersection exist between IS and CS curricula that may be exploited in order to teach to both audiences. The course was developed with an eye toward minimizing costs and faculty preparation in addition to utilizing existing laboratory facilities. The course concept has been determined to be scaleable for large and small institutions as well as applicable to the needs of both IS and CS curricula. An unexpected benefit of the course is the many opportunities found for integrating and reinforcing concepts found across computing curricula. Finally, we have included our software toolset (see Appendix), which will assist in the development of telecommunications laboratory experiments that reinforce the course content.

5. ACKNOWLEDGEMENTS

The authors wish to thank Rainey Little and Chuck Thomas for their helpful suggestions and encouragement.

6. REFERENCES

ACM/IEEE-CS Joint Curriculum Task Force, Computing Curricula 1991, 1991 http://www.computer.org/education/cc1991.

AITP, IS'97 Curriculum Model for 4 Year Undergraduate Programs in Information Systems, 1997,http://www.is2000.org/is2k/rev/Review1.asp.

Battig, Michael E., "Scaling a Data Communications Laboratory Course to Fit the Small College Curriculum," 1998, The Journal of Computing in Small Colleges, vol. 13, no. 4.

Harrington, Jan L. and Helen M. Hayes, "A Joint First Year Program for Computer Science and Information Systems," 1995, SIGCSE Bulletin, vol. 27, no. 1, pg. 121-5.

Johnson, Hubert A., "Integrating Software Engineering into the Traditional Computer Science Curriculum," 1997, SIGCSE Bulletin, vol. 29, no. 2, pg. 39-53.

Nelson, Mark, Serial Communications: A C++ Developer's Guide, 1992, M & T Publishing Inc.

Shay, William A., Understanding Data Communications and Networks, 2nd Ed., 1999, PWS Publishing Co.

Smith, Wayne, "Tutorial: A Laboratory to Support a First Course in Data Communications Using Personal Computers and Turbo Pascal," 1994, SIGCSE Bulletin, vol. 26, no. 1, pg. 404.

Smith, Wayne, "The Design of An Inexpensive Undergraduate Data Communications Laboratory," 1991, SIGCSE Bulletin, vol. 23, no. 1, pg. 273-6.

Stallings, William, Data and Computer Communications, 6th Ed., 2000, Prentice-Hal, Inc.

Tannenbaum, Andrew S., Computer Networks, 3rd Ed., 1996, Prentice-Hall, Inc.

APPENDIX

Four complete Visual Basic 5.0 programs are provided in our zip file. To download the zip file, visit the web site at: academics.smcvt.edu/compsci/CSLinks.htm. The four programs are described below along with details about the laboratory assignments.

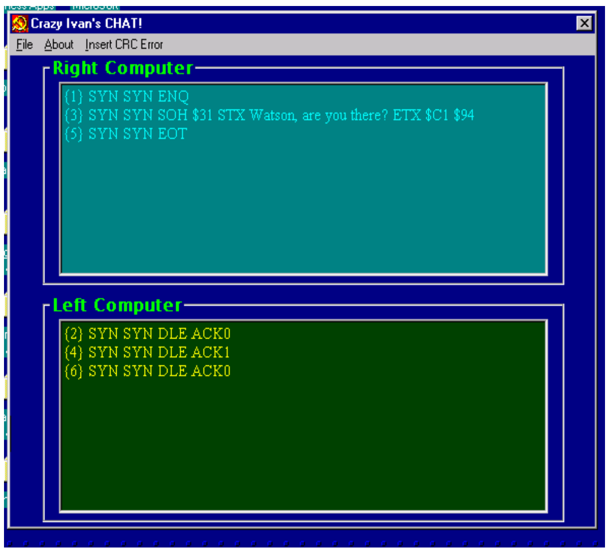
The Monitor program is perhaps the most critical piece of software for the laboratory experiments. Both students and faculty need the Monitor program for program validation. We recommend that students not be given access to the Monitor source code since it contains too much that could be pirated for the lab work. An actual run of the Monitor program is shown below.

The Chat program is a simple communication program that demonstrates the use of UDP (Winsock 5.0), Visual Basic, and the Monitor to establish connections between two workstations without any particular protocol or data frame (i.e., typed characters are simply transmitted as they are keyed and received characters are displayed in a separate window). The Chat program's source code serves as the starting point for students constructing their own programs that will implement the BSC protocol. Therefore, the Chat program is the only program for which the students receive the source code.

The CRC program is a standalone program that calculates CRC-16 values for input character strings. This program serves two purposes. First, it provides a source for validating computed CRC values by students. Second, for those faculty members who do not wish to inflict students with the pain and suffering of building their own CRC routines, this source code may be provided to students. In this second case, the students will still have to incorporate the logic into their programs to send/receive NAKs when the Monitor corrupts CRC values.

The final program is the complete DataComm program that can send, receive, calculate CRC values, and recover from errors injected by the Monitor program. Care must be taken to prevent students from obtaining the source code of this program. This program serves two useful purposes for students. First, it provides them with a working model of how their finished program should function. Second, it provides the necessary sending capability that they need to build a receiver program (the very first assignment).

Finally, it is worth mentioning that these programs could quite possibly be adapted to work with other frame formats or protocols. For example, one could construct a circular sequence of stations and implement a token ring laboratory.



```
{1} SYN SYN ENQ
{3} SYN SYN SOH $31 STX Watson, are you there? ETX $C1 $94
{5} SYN SYN EOT
```

```
{2} SYN SYN DLE ACK0
{4} SYN SYN DLE ACK1
{6} SYN SYN DLE ACK0
```

A sample execution of the Monitor program (written in Visual Basic 5.0)