Int. J. Engng Ed. Vol. 11, Nos 4 and 5, pp. 306–313, 1995
Printed in Great Britain.

0949–149X/91 $3.00+0.00

An Undergraduate Data Communications Laboratory

WAYNE D. SMITH
Department of Computer Science, Mississippi State University, MS 39762, USA

This paper reports on the design, implementation and operation of an undergraduate data communications laboratory over a four-year period at Mississippi State University. This laboratory was developed with support from NSF and contains 10 IBM-compatible microcomputers. Communications are accomplished through the serial ports of the computers using RS232 protocols. A series of experiments was designed that give students practice in writing communications protocols in the Turbo Pascal language. A significant quantity of software was developed locally to support this laboratory.

AUTHOR QUESTIONNAIRE

1. The paper describes new training tools or laboratory concepts/instruments/experiments in:
   A first course in data communication and/or computer networks.
2. The paper describes new equipment useful in the following courses:
   The equipment is not really new. Standard personal computers and the Turbo Pascal programming environment are used in a new approach to the teaching of computer data communications through the use of a planned sequence of programming assignments implemented in a data communications laboratory.
3. Level of students involved in the use of the equipment
   Upper-division undergraduate and graduate-level students are and have been involved with this equipment in this laboratory.
4. What aspects of your contribution are new?
   The use of a laboratory to support the teaching of data communications concepts is relatively new. Also, the concept of having the students actually accomplish the programming of data communications protocols in data communications laboratory is new.
5. How is the material presented to be incorporated in engineering teaching?
   An instructor reading the paper could pretty much establish a similar laboratory with little additional information. However, supporting materials, including class handouts, programming assignments, laboratory protocols and supporting software, are available from the author without cost. Using this material would greatly facilitate the establishment of such a lab.
6. Which texts or other documentation accompany the presented materials?
   The two textbooks that I have used to support this course are:
   William Stallings, *Data and Computer Communications* [1].
   Andrew S. Tannenbaum, *Computer Networks* [2].
   I am currently using the fourth edition of the Stallings book. In addition, a number of class handouts are used to present the details of the protocols used in the class. These are available free of charge from the author.
7. Have the concepts presented been tested in the classroom. What conclusions have been drawn from the experience?
   Yes, I have used this laboratory for four years. The results have been excellent.
8. Other comments on benefits of your presented work for engineering education.
   Engineering education at almost all levels is greatly enhanced by the use of laboratory experiences used to reinforce the materials learned in the classroom. This laboratory was established in an effort to provide the students with a better education in computer communications than is possible in any survey course without a laboratory. A number of other institutions have requested materials for use in setting up similar laboratories of their own.

INTRODUCTION AND HISTORY

IN THE FALL of 1987, the Department of Computer Science in conjunction with the Department of Electrical Engineering at Mississippi State University introduced an undergraduate course in data communications and networks that would become a required course for computer engineering majors. This course was introduced primarily to provide computer science and computer engineer-

ing graduates with a firm background in data communications.

Based on the conviction that a laboratory to support this course was essential for a thorough coverage of the material, a project was initiated in 1987 to develop a laboratory to support the proposed course. Between 1987 and 1989, the specifics of the requirements for such a laboratory were developed, along with a basic outline of the laboratory exercises that would be used in the course.

As the plans for the laboratory were being developed, a proposal was submitted to the National Science Foundation to obtain a matching grant under the Instrumentation and Laboratory Improvement (ILI) program. This grant was approved in March of 1989, and the laboratory equipment was installed in the Fall of 1989.

THE PHILOSOPHY BEHIND THE LABORATORY

The major goal of the project was to establish the undergraduate course in data communications and to develop the undergraduate data communications laboratory to support classroom instruction. The major consideration in structuring the laboratory was to provide the students with exposure to data communications in more depth than at just the user level. The experiments and laboratory equipment were intended to provide the students with the practical experience needed to prepare them to assume a role in computer data communications systems design and implementation.

Because of the need for this designer-level interaction with the network, the idea of using a laboratory equipped with any of the many off-the-shelf networks was rejected early in the design phase. It was felt that while a laboratory of this type would provide the student with some facility for using a computer network, it would not provide the hands-on experience at the programming level that was needed for the students in question. Using a commercial network would, in fact, mask the very features that the data communications course was being designed to address.

It was also decided that the laboratory experiments should be built on the students' prior course experiences and should not require the learning of new computer languages or operating systems. Since all computer science, computer engineering and electrical engineering students at Mississippi State take a course in Pascal, this language was chosen as the programming tool for the laboratory.

In designing the laboratory experiments, it was felt that no single network topology would address all the issues that needed to be covered in the course. For this reason, it was decided that the laboratory should be designed to permit rapid reconfiguration of the network being used. This required a collection of machines of very similar architecture that could be treated as identical modules when the network topology was being configured. In addition, facilities would have to be provided so that very rapid changes in interconnections would be relatively easy to accomplish.

Finally, it was desirable to keep the cost of the laboratory to a minimum. This meant that the number of computers and the cost per machine should be kept as low as practical.

THE LABORATORY EQUIPMENT SPECIFICATIONS

Based on the preceding specifications, the decision was made to equip the laboratory with IBM-compatible personal computers and to implement all the data communications operations through the serial ports of these machines. The primary attraction of this type of machine is its low cost. Another advantage of the IBM clones is that the students are familiar with this type of computer and its operating system. In addition, the majority of these machines come with two serial ports, and additional ports can be added at a relatively low cost. Further, most of the students taking the course are familiar with Turbo Pascal, which runs on these machines.

To facilitate the reconfiguration of the topologies of the networks for different experiments, modular phone plugs were used. An RS232 to modular plug adapter was used that permits interconnections between the various machines using four-conductor modular phone cables and plugs. A modular cable is run from each serial communication port on each PC to a modular plug patch panel. From the patch panel, the communications ports on one PC can be connected to the communications ports from another PC by using an additional modular phone cord. The computers purchased for use in the laboratory include two serial ports. The use of the patch panels and modular phone cords permit the establishment of a wide variety of network topologies that are useful in the laboratory. The reader who is unfamiliar with data communications is referred to the textbook by Stallings [1] or an equally good text by Tannenbaum [2] for introductory information. Additional information is included in the glossary at the end of this paper.

Figure 1 indicates how a point-to-point topology is accomplished. This arrangement provides full duplex communications between the two machines. If needed, half-duplex operation can be simulated by having a transmitting machine empty the receive buffer of any data that arrives during its transmission period.

A token ring network [3] can be configured as shown in Figure 2 by successively interconnecting the communications port 1 on one machine to communications port 2 on the next machine until a ring is completed. Each station transmits on one port and listens on the other port. Hughes [4] suggests that a bus topology (carrier sense, multiple

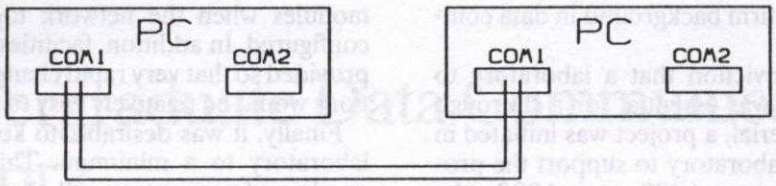

Fig. 1. Point-to-point topology.

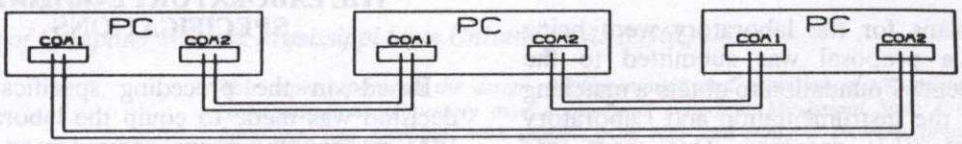

Fig. 2. Ring or bus topology.

access with collision detection—CSMA/CD) [5] can be simulated by utilizing a structure similar to the token ring network. Each station transmits on one port and listens on another. A station initiates a transmission only after listening to ensure that the bus is not in use. Any non-transmitting station that receives traffic on its input port automatically relays that traffic to its output port.

The transmitting station listens to its receiving port while it is transmitting. If the received data does not match the transmitted data, the transmitting station assumes that another station is also transmitting and that a collision has occurred. Stations involved in a collision immediately stop transmitting and use a random backoff algorithm to establish a time at which to resume transmission.

The serial communications port approach to data communications is most appropriate to character-oriented protocols involving asynchronous communications. However, it is not limited to this type of protocol. By treating the lower-level procedures as service access points (SAP) for the higher layers, the details of the asynchronous data communications are effectively hidden from the routines written by the student. Therefore, the system can be thought of as either a synchronous or an asynchronous system. Implementing a bit-oriented protocol can also be accomplished through the serial ports. To accomplish this type of operation, a full byte is transmitted, but only 1 bit of the byte is treated as data by the sender and receiver. This 'fat bit' protocol was used by Margaret M. Reek [6] in a serial input/output-based data communications laboratory utilizing SUN workstations.

At Mississippi State, 10 computers are used to support the communications laboratory. This is an adequate number to support an enrolment of about 20 students in a laboratory section. This provides five pairs of computers during point-to-point experiments. It also allows for two or more ring or bus topologies, as needed. The microcomputers have 30 Mbyte hard disks and parallel printers.

After several semesters of trial and error, the configuration that has evolved is one that contains three clusters of machines. Two clusters contain three computers, and the fourth contains four machines. Each cluster has its own individual patch panel and a collection of interconnecting wires. This configuration permits three groups of students to be at work simultaneously. Three machines provides for a sender, a receiver and a network monitor if needed. If larger networks are needed, longer modular cables can be used to interconnect the clusters. The laboratory has supported a network of up to eight machines at one time.

Recently the laboratory equipment has been augmented with several Sperry portable personal computers that became available to the department without cost. These machines are used exclusively as system monitors, and free up the hard-disk, 80386 machines for programming and data communications experiments. It represents an excellent way to utilize some equipment that would otherwise have been discarded.

LABORATORY SOFTWARE FACILITIES

Each computer in the lab is provided with DOS and a copy of Turbo Pascal. Originally, the laboratory used Turbo Pascal version 3.0, but this has lately been upgraded to version 6.0. On each computer, there are two subdirectories that contain software that has largely been developed locally for use by the students.

In one subdirectory is a collection of routines that can be invoked from within the student's Pascal program to provide the basic physical layer functions required for the data communications operations. The operations provided by these routines include configuring the communications ports, reading a byte from the communications port, and writing a byte to the communications port.

Additional software tools that are used to assist

the student with programming assignments are provided in a second directory. These tools are generally object versions of software that can be executed, but not read or modified, by the student. Two examples are the Xmit program used as a transmitter while the student is writing the receiver program, and the Monitor program that permits the student to observe traffic between computers during testing and debugging.

HOW THE EQUIPMENT IS USED

The equipment in this laboratory is used to explore various levels of telecommunications operations through a series of experiments involving data communications hardware and software. The students begin by writing a very simple point-to-point data receiver program that uses a very simple protocol. Throughout most of the remainder of the semester, the student expands and modifies the program from the preceding experiment in order to provide additional features and services to the communications protocol. This point is stressed at the beginning of the semester, and the students are encouraged to write modular, structured code that conforms to the general Open Systems Interconnection (OSI) seven-layer communications model. The students who produce the better code in the beginning reap the rewards in later experiments.

About one-half of the semester is devoted to basic telecommunications experiments. These experiments utilize pairs of microcomputers connected in a point-to-point configuration. Within this topology, the students implement a simple half-duplex character-oriented protocol that approximates a subset of IBM's Binary Synchronous Communications (BSC) protocol, which we call Baby BSC or BBSC. During these experiments, the students generally work in groups of two to facilitate the operation of two machines at once.

During the first five experiments, the students are required to build some debugging aids into their programs. Beginning with assignment 2, the students use an 'Alt E' key routine that is built into their programs to turn a 'message echo' feature on or off. When the echo mode is on, every character that is transmitted or received is also displayed on the screen, using separate windows for transmitted and received data.

The first laboratory experiment involves the computation of a simple 2–byte cyclic redundancy code (CRC) value computed for a text string input. This experiment gives the student some experience with logical and arithmetic operations on byte and bit type variables, and the CRC routines written in this experiment are later incorporated into the data link layer in future communications experiments.

In the second experiment, the student is given a working copy of an object program (Xmit) that will accept a text message from the keyboard and then put the message into the BBSC format and transmit it over the communications port using the full BBSC protocol. The student is required to write a receiver that will receive such messages and display them on the screen of the computer. Text messages are limited to 80 characters for this experiment. If the Echo mode is enabled, the student will be able to view the full exchange of BBSC packets as the communications take place.

In experiment 3, the students write a BBSC transmitter procedure and integrate this program into the receiver from the previous experiment. At this point, the students have a working communications protocol that will enable pairs of machines to exchange messages.

In experiment 4, the students add error detection and an automatic repeat request (ARQ) error correction protocol to their receivers. The monitor program that the students have used in debugging their earlier programs is used to inject errors into the transmission stream.

Experiment 5 consists of bidirectional ASCII text file transfer. The student enters the file name and the file is transferred as a series of 80 character blocks. Because of the limited nature of the BBSC protocol, only ASCII file transfer is supported. The students demonstrate the correct transfer of a file by transferring a Turbo Pascal file that must then be compiled and executed in order to receive credit for this assignment.

The file transfer program uses block numbers as a type of sliding window protocol to prevent the reception of a duplicate data block in the event of a lost acknowledgement (ACK). The file transfer program is first tested without errors and then with errors.

Experiment 6 is one of the more interesting experiments. Throughout the semester, all students have been using the same protocol specifications, but have only tested their program's communication capability with another copy of the same program. In experiment 6, each student group tests their program for operation with each other group's program. A full class period is devoted to this experiment, and all the students assemble in the laboratory for the test session. It generally generates a lot of excitement from the students, and no small amount of controversy as to the exact meaning and interpretation of some of the protocol specifications.

After the first six experiments, the students are assigned to a larger group of from three to five students. These groups then work on a single large programming assignment that repeats most of the previous experiments, but this time using a more complex network topology. This assignment is graded in two or three phases, involving simple e-mail type text transfer, file transfer and error recovery. A minimum of three machines must be used, which imposes addressing considerations in the network layer.

In the larger project, generally either a ring or a bus topology is used. Wiring the computers into a ring topology will support the use of a token ring

protocol and can also be used to simulate the operation of a bus or CSMA/CD protocol as described earlier.

Whichever approach is used will provide the student with the opportunity to implement some parts of the network layer of the OSI model. The protocol implemented will vary, depending on the medium access protocol chosen [7]. To date, topologies employed in the lab have consisted of token ring, star and bus networks. Within the token ring topology, both byte-oriented and 'fat bit' techniques have been used.

Unfortunately, it is not possible for each student to cover all these topics in a single semester. In any one term, all students cover the basic point-to-point experiments. The student groups are then either assigned a topology or allowed to chose one of particular interest to their group. The group then works on that one network for the remainder of the semester. Not all groups always work on the same network for the project.

The testing and grading of the student programs has turned out to be something of a challenge. The programs written by the student are sufficiently long that just reading the code to determine correctness is not feasible. Therefore, each student group's program must be tested on-line in the laboratory. This is very time consuming, but has proven to be the only practical method for determining correct operation. Specific testing procedures have been developed for each assignment, and the monitor program has been very useful during these grading tests. The students are also required to turn in a hard copy listing of their program for examination, but the majority of the grade is determined from the live test.

DEVELOPING SOFTWARE SUPPORT FOR THE LABORATORY

When the course was first developed, the students were supplied with the source code for a set of Pascal subroutines that would perform most of the OSI Physical Layer functions of the data communications protocol that was to be used. These routines were in the form of files that could be integrated into the student programs at compile time. The software included routines that would initialize a serial port, read a byte from a serial port and write a byte to a serial port. The code also included an interrupt-driven data receiver program, written primarily in in-line machine code.

After the course was taught the first time, it was determined that the students would need considerably more support if the laboratory was to be effective. This would include a modification of the assignment structure and the provision of additional software support.

A major improvement in the laboratory procedure involved breaking the second assignment into a transmitter and a receiver section, with the student working on one section at a time. To permit the student to develop a receiver, however, requires access to a transmitter program with which to communicate. To support this approach, a transmitter program called Xmit was developed and provided to the students in the form of an object file. The use of this program and approach greatly increased the percentage of students who were able to successfully write the code to develop the receiver program.

Once the student has a working receiver program, the next experiment requires the addition of a transmitter function to the same program. This modification to the student's original program introduces a measure of uncertainty into the debugging process. When an error in communications occurs, the most common indication is that one or both the computers hangs in some undetermined state, with little or no indication as to what the problem might be. During the first course, this situation pointed out the need for a monitor program.

A monitor program was designed and written so that a third computer could be inserted into the physical link between two machines that were involved in the communications process. The monitor software permits the third computer to function essentially as a 'display and forward' device. The data that was received on one serial port was automatically transmitted on the second port, and vice versa. At the same time, two windows are used on the terminal screen to display the data as it is received on the two different communication ports. The monitor software was written so as to recognize and respond to the point-to-point protocol that was in use. That is, the monitor would print appropriate text strings to represent the control characters that were used in the protocol. It was necessary to convert the control characters to a text string equivalent since most of the control characters used in the BBSC protocol cannot be displayed on the terminal screen.

The initial use of the monitor was limited to monitoring and displaying the communication messages that passed through the communications link. Once the monitor was placed into the communications link, however, it became clear that the monitor could also be used to intentionally inject errors into the data stream. This provided the student with a realistic source of errors to use in testing error detection and correction procedures.

The monitor uses a random number generator with a variable parameter that is set by the user in order to generate errors in some desired proportion of messages. Two different types of errors can be generated in the monitor. These errors are a CRC error, which represents one or more bit errors in a transmitted message, and a missing acknowledgement error, which represents a lost message. Either error percentage can be set from the monitor keyboard at any value between 0 and 100%.

Based on the percentage entered from the key-

board and a value obtained from the random number generator, the monitor periodically changes the last byte of the CRC to a value of 0x00. This produces the same results as a corrupted bit in the data stream, since the CRC value contained in the frame is no longer appropriate for the text message that is in that frame. The station receiving a message containing an incorrect CRC value sends a negative acknowledgement (NAK) to the originator of the message to request that the message be retransmitted.

The missing acknowledgement process operates in much the same fashion. The user sets the percentage of acknowledgements that are to be lost. Based on this percentage and a random number, the monitor will periodically fail to forward a received acknowledgement. The program that is expecting this acknowledgement will detect the missing acknowledgement through the use of a 'time-out' timer. If the program does not receive the expected acknowledgement within 3 sec, the program must take corrective action.

Both the CRC and ACK errors can be enabled at the same time. During the evaluation of student programs, the programs are tested with a simultaneous error probabilities of 40% for both the CRC and ACK errors. This test has been very effective in demonstrating to the students that data communication is possible with a simple protocol with limited error correction capabilities under adverse physical channel conditions.

In 1991, Turbo Pascal 6.0 became available in the Department of Computer Science, and a migration of the laboratory software was initiated to the new system at that time. The most immediate improvement in the laboratory software under the new programming environment was the change in the communications port handling software. Since the new version of Pascal permitted access to the system hardware at a much lower level than had the earlier system, this permitted the writing of interrupt routines in Turbo Pascal itself, and thus eliminated the need for the extensive machine language routines that had been used in the earlier software. This had the advantage of providing the students with software that they could read, understand and modify. This software has proven to be much more robust than the original machine code.

When the transmitter and monitor programs were converted to Turbo Pascal 6.0, a modification was also made in the protocol that permitted the instructor to induce some variation into the frame format of the basic communication block. This change in the two programs resulted in giving the instructor the capability to change the frame format from semester to semester. This was desirable as a method for reducing the 'assignment spillover' from one semester to the next. The Xmit program was modified to request the header format, and then the program uses the header format supplied by the user. When the monitor was rewritten, it was redesigned to accept a variety of header formats without input from the user.

Finally, the conversion to Turbo 6.0 was also accompanied by the development of a monitor program for the token ring protocol. The token ring monitor was written so as to include an error injection capability. Error injection in the token ring environment is somewhat more complex than in the point-to-point case. In any general token ring with multiple stations, the position of the monitor in relation to the sender and receiver of a message cannot be determined in advance. For this reason, the monitor must inject both a CRC and a NAK into a single frame to ensure that at least one of these abnormal conditions returns to the originator of the message. Since either or both of these conditions will result in the retransmission of the original packet, both the sender and receiver have to deal with retransmission. The probability of an error can be set on the token ring monitor in a manner similar to that used with the point-to-point monitor.

SUMMARY AND RESULTS

The laboratory discussed above was installed at Mississippi State University in the Fall of 1989. Some use was made of the laboratory with a graduate class in the Fall of 1989, and the first undergraduate class of some 20 students used the facilities in the Spring of 1990. Since then, the laboratory has been used every semester, with graduate and undergraduate data communications classes offered in alternating semesters.

Student feedback indicates that the laboratory assignments are challenging, and they report that the learning experience is quite worthwhile. In addition, several students have reported that they found the programming experiences in the laboratory to be enjoyable. They have stated that they found the debugging of the programs to be frustrating due to the real-time nature of the programs, but they felt that such programming provided an excellent learning experience. Several students have also stated that they have found that the evolving nature of the programming assignments have provided a valuable adjunct to their software engineering studies.

Interest from the academic community has also been good. To date, approximately 12 institutions, including one from Poland, have requested additional details about the course and the laboratory. Course materials, supporting software and other information have been provided to those schools. In addition, the author presented a tutorial on the features of this laboratory at the Special Interest Group on Computer Science Education Symposium in March of 1994.

The original software has now been in use for four years, and the new software has been in use for two semesters. The results have been excellent. The percentage of students who get the communications programs operational by the deadline has increased steadily, and is currently somewhere

above 90%. An occasional due-date extension has been necessary, but the majority of the students complete the assignments on time. This success rate is very important in this course, since the modular nature of the assignments makes it imperative that experiment n be completed successfully before experiment $n+1$ can be undertaken.

An unanticipated benefit of the modular assignment approach became evident after a couple of semesters. This is probably the first course where the students were required to work on a single assignment through five or six assignment modifications. This approach has resulted in an increased appreciation of structured programming on the part of the students. Even with good programming style, the final point-to-point program is usually pretty patched up, and without good technique, it can be a hopeless mess. Either way, by the time the last point-to-point assignment is completed, the students are usually quite anxious to abandon the point-to-point protocol and begin work on the token ring network.

Another unexpected discovery in the laboratory was that the students seem to encounter significantly less difficulty in programming the token ring assignment than they do with the same features in the point-to-point environment. This is true even though the token ring involves at least three stations, and the students have not usually had a monitor to help with the debugging of the token ring. The students also usually express more enthusiasm and interest for this part of the laboratory assignments. It is unclear why the students do better with this assignment. It may be that the fundamental concepts of a token ring protocol are intuitively easier to grasp. The instructor would like to think, however, that the experiences gained in the point-to-point experiments result in a positive transfer of concepts that can be applied to the token ring system. Whatever the reason, the students are normally able to complete in 2–3 weeks on the token ring the same level of data communications facilities that they completed in 6–8 weeks on the point-to-point experiments.

Overall, the results from the laboratory have been very encouraging, and the laboratory appears to meet all expectations. It provides a low-cost method for providing students with the hands-on experiences needed to prepare them for employment in a computer industry where a knowledge of data communications is becoming increasingly important. Results have been so good that a second course in data communications has now been developed based on the Internet protocol, and was taught for the first time in the Fall of 1993. Students are now beginning to ask for a third course in this area.

Acknowledgements—This material is based in part upon work supported by the National Science Foundation under grant no. CDA-8852902, and the US Government has certain rights in this material. This grant provided one-half of the funds required to equip the laboratory with hardware, and the facilities provided in this lab would not have been possible without NSF support. Most of the software used in the laboratory was developed by students at Mississippi State University. Some of the initial serial port software came from public domain sources, with some modifications by the author. The original programming environment support software was developed by John Rutledge, and the first communications port support software in Turbo Pascal 6.0 was developed by C. Lakey, B. Williams, S. Canfield and I. Hudson. The rewrite of the remainder of the support software in 6.0 was the work of James Dement. A large measure of thanks is owed to all these people. Since the software was developed as an means to support the teaching of computer data communications software, it is readily available to all who would like to have a copy. Both source and object files are available for all the software mentioned above. To obtain this software, contact the author by the most convenient means. Supporting documentation including a course syllabus and programming assignments is also available.

REFERENCES

1. W. Stallings, *Data and Computer Communications*, 4th edn, Macmillan, New York (1994).
2. A. S. Tannenbaum, *Computer Networks*, 2nd edn, Prentice Hall, Englewood Cliffs, NJ (1988).
3. IEEE Standards Board, *An American National Standard IEEE Standards for Local Area Networks: Token Ring Access Method and Physical Layer Specifications* (1985).
4. L. Hughes, Low cost networks and gateways for teaching data communications, *SIGCSE Bull.*, **21** (1), 6–10 (1989).
5. IEEE Standards Board, *An American National Standard IEEE Standards for Local Area Networks: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications* (1985).
6. M. M. Reek, An undergraduate concentration in networks and distributed systems, *SIGCSE Bull.*, 21, 12—16 (1989).
7. IEEE Standards Board, *An American National Standard IEEE Standards for Local Area Networks: Logical Link Control* (1984).

GLOSSARY

ACK (Acknowledgement)—a positive acknowledgement used with an ARQ protocol.

ARQ (Automatic Repeat Request)—a protocol that uses positive and negative acknowledgements with retransmissions to accomplish reliable communications.

BSC (Binary Synchronous Communication)—an early IBM protocol for transmitting data across a synchronous communications link.

BBSC—a locally developed subset of BSC proto-

col used at Mississippi State University in the data communications laboratory to accomplish point-to-point communications.

CRC (Cyclic Redundancy Code)—a technique that adds additional bits to a transmitted message with the object of detecting errors that may occur during transmission of the data.

CSMA/CD (Carrier Sense, Multiple Access with Collision Detection)—a network access protocol where a station with traffic, first determines that another station is not transmitting and then transmits data; if a collision is detected during the transmission, the station stops transmitting, and tries again later.

NAK (Negative Acknowledgement)—used with the ARQ protocol to indicate an unsuccessful data transfer.

OSI (Open System Interconnection)—a set of protocols specified by the International Organization for Standards to specify methods for interconnecting communications units.

RS232 (Recommended Standard number 232)—an Electronic Industries Association standard for interfacing computers and terminals to the public telephone system for the purpose of data transfer.

SAP (Service Access Point)—the vehicle through which one level of a communications protocol communicates with the level above and below that level.

Sun—a brand of computer workstation closely associated with communications networks.

A native of Middlesborough, KY, Dr **Wayne D. Smith** received a BS from Auburn University in 1969, an MS from Georgia Tech in 1970, and a Ph.D. in computer science from the University of Illinois in 1976. He is currently a professor in the Computer Science Department at Mississippi State University. For several years, his educational interests have involved the design and development of laboratories to support undergraduate education in computer science and engineering. He has been the recipient of two NSF grants for the establishment of undergraduate teaching laboratories, one in computer organization, and one in data communications. He also developed a microcomputer design laboratory that was used in teaching microcomputer architecture at NASA Johnson Spacecraft Center and at White Sands Missile Range. His current teaching responsibilities are in data communications and networks, and his research activities have included contracts with NASA and the US Air Force.