# Simulation Game for Teaching Communication Protocols

Eyal Shifroni
Center of Educational Technology (CET)
16 Klausner St.
Tel-Aviv, Israel 61394
email: eyal_s@mail.cet.ac.il

David Ginat
Science Teaching Department
Weizmann Institute of Science
Rehovot, Israel
email: ntginat@weizmann.weizmann.ac.il

## Abstract

During the development of a new program for teaching the subject of computer communication at high school, we examined different ways of teaching the algorithmic aspect of communication protocols. Upon trying the "standard" frontal lecture-type presentation, only a minority of the students comprehended the protocols.

In looking for an alternative way of teaching, we developed a simulation game in which the students act as the protocol components. After playing the game, they discuss the characteristics of the protocol as experienced through playing.

We found the simulation game method to be far more effective than the lecture-type presentation. The simulation game method had significantly improved the level of understanding and the motivation of the students.

In this paper we describe the simulation game and its didactic method and discuss the results of applying it in high school classes in Israel during the past year.

## Introduction

During the last three years, the Center for Educational Technology and the Ministry of Education in Israel have developed a new program for teaching the subject of computer communication in high-schools [10]. The program is intended for students in twelfth grade, majoring in Information Technology (IT). The program is meant to reflect the significant developments that has occurred in this field in recent years, and to demonstrate the growing importance of the area of computer communication.

In this new program, a considerable emphasis is put on understanding the structure and the behavior of basic communication protocols. The rationale for this emphasis is that IT majors *use* communication networks, and should know and understand *design* considerations of these networks. We believe that studying the algorithmic aspect of communication protocols is the proper way to give the students insight into the problems and the tradeoffs involved in the design of computer networks.

SIGCSE '97 CA, USA
© 1997 ACM 0-89791-889-4/97/0002..$3.50

During an experimental application of the new program, we found that the task of teaching even the most basic protocols to high-school students is not easy. The students, who had thorough familiarity with sequential programming techniques, had considerable difficulties in understanding distributed algorithms. In particular, they failed to realize the various sequences of events that can occur in a communication system and must be handled by a communication protocol.

The students' difficulties are actually not surprising. Communication protocols are complex algorithms which have characteristics that are unfamiliar to high school students: they are *distributed*, *concurrent* and they include *uncertainty* (since it is never certain when, and if at all, a message sent from one end of a communication link will reach the other end).

Teachers that tried to teach the protocols in a frontal lecture-type explanation reported that the majority of the students had severe difficulties in understanding the distributed and concurrent occurrences. In particular, the students found it hard to understand handling of message loss, message reordering and message duplication.

The frontal lecture format does not address the need of students whose learning tendency is more on the active side of the active-reflective dimension in Felder's learning model [4]. It became rather clear that some kind of active illustration was required.

Previous work has shown that illustrations can enhance the learning of algorithm behavior. Illustrations were provided through computerized animation, visualizations, and games with objects. Computer science educators have used algorithm animation and reported that this technique assists students in understanding various algorithmic aspects, including complexity [e.g. 2,9]. Although animation could assist in teaching communication protocols, it is still somewhat abstract for high school students. In addition, the students are required to learn how to use the animation system.

Harvey [6 ] used a visualization technique which he calls "the little people metaphor" to explain subjects like procedure calls, parameter passing and recursion. This metaphor was used successfully in teaching these concepts to high-school students [7 ]. Chung et al [3], Ginat [5], and others have used games with physical objects for teaching scientific and algorithmic principles. Game activities are attractive, facilitate classroom interaction, and allow whole class participation.

Our idea was to combine game activities with the metaphor of the "little people" by letting the students act out the protocol. In the game, students take the role of the "little people", and simulate the behavior of the protocol. Such a simulation is appealing, especially for high-school students. In this paper we focus upon Data-Link protocol game, and describe how the problems and the considerations of the Data-Link layer were realized through playing the game.

## The Stop & Wait Protocol

The basic protocol of the Data-Link layer is the stop & wait protocol. In order to understand our simulation game it is important to be familiar with this protocol. In this section we briefly introduce basic considerations of the protocol.

The Data-Link layer is the second layer (out of seven) in the OSI reference model[1]. It provides services to the layer above - the Network layer, and uses the services of the layer below - the Physical layer.

In a communication system, information is transmitted between nodes. We examine a half duplex link, in which one node is a *sender* and the other is a *receiver*. The *Network sender*, namely, the Network layer component in the sender node, divides the information for transmission into packets, which are sent to the *Network receiver*.

The packets transmission is handled by the Data-Link and the Physical layers. The *Data-Link sender* adds control information to each packet and creates a *frame*. *The Data-Link receiver* peels off the control information and uncovers the original packet. The Physical layer transmits frames over the physical channel connecting the nodes.

Noise in the channel can cause errors in frames or loss of frames. Heavy load in the channel can cause unlimited transmission delays. The control information in the frames is used by the Data-Link layer for handling possible disruptions of frame transmission at the Physical layer.

[1] The OSI model is covered in most textbooks dealing with the subject, see for example [11].

The task of a *Data-Link protocol* - the protocol between the Data-Link sender and receiver - is to ensure that all packets sent by the Network sender will reach the Network receiver: 1) in-sequence, 2) each packet exactly once and non-corrupted.

*The stop & wait* protocol is a Data-Link protocol designed to perform this task. This protocol requires that the (Data-Link) sender will stop after the transmission of each frame, and wait for a response from the receiver. If the response is negative (Nak), the sender will retransmit a copy of the frame. If the response is affirmative (Ack), the sender will send the next frame.

## Teaching the Stop & Wait Protocol

The teaching methodology we developed consists of the following three steps:

A. Introduction of the basic principle of the protocol, discussion of problems that may arise in its operation, and suggestion of solutions.
B. Playing the simulation game, and testing and debugging the protocol.
C. Reviewing the lessons learned from playing the game, and analyzing the efficiency of the protocol in terms of link utilization.

Each of the steps is explained in detail below.

### Step A. Introduction of the Protocol

The basic principle of the stop & wait protocol must be introduced before the students are able to simulate the protocol. The students are first presented with the possible disruptions at the Physical layer. They are then taught the use of *checksum* for detecting corrupted frames. Finally, they are asked to answer the following three questions (preferably without, but sometimes with, guidance from the teacher):

Q1. How can the protocol overcome corrupted frames detected by the receiver?
The answer to this is that upon detecting a corrupted frame, the receiver sends a Nak response, which signals the sender to retransmits the original frame.
Q2. How does the protocol handle a lost frame?
If a frame is lost, a response for it will never be received, hence the sender must start a timer upon sending each frame, and retransmit the frame if the time runs out.
Q3. How does the protocol handle a lost response?
This is a more subtle question. When a response is lost, the sender will time out and retransmit the frame. But if the frame has already been received successfully, it would be received twice. To handle such duplications, the sender must number packets, and the receiver must ensure that packets are accepted without duplications.

### Step B. Playing the Game

After the students have understood the way the protocol handles the communication problems discussed above, they are ready to start the game. The game requires five groups. These groups and the transfer of information between them are depicted in figure 1.

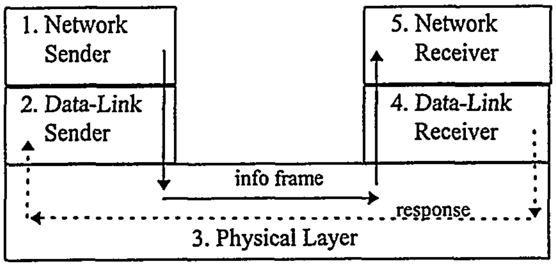

Figure 1. Groups and information transfer in the game

The sending and receiving groups are located in separate rooms, and can communicate only through the services of the Physical layer group. The teacher gives the Network sender group a short message, consists of 3-4 sentences, and this group should use the services provided by the Data-Link sender group, in order to pass the message to the Network receiver group. The message constructed by the Network receiver group should be identical to the message originally given by the teacher.

**Illustration Means**

Upon developing illustration means, two questions arose:

1. How to simulate corruption of frames?
2. How to simulate delays and losses of frames in the channel?

Corruptions, delays, and losses occur randomly, hence some random device should be used. We chose a simple random number generator: a die.

For illustrating a frame we used the following form:

| seq. num. | information | ctrl |
|---|---|---|
| | | |

For illustrating the variables of the protocol, we used pieces of paper.

**Instructions for the Game**

The actions of the five groups are described bellow[2].

I. The Network sender group:

Divides the message given by the teacher to short pieces, called *packets*, and passes them to the Data-Link sender group, one by one - when required.

II. The Data-Link sender group:

1. Writes on a piece of paper: '<u>SN</u> : 0'. The paper represents a Sequence Number variable, SN, which is initialized to 0.
2. Receives the next packet from the Network sender group.
3. Fills in a form by: writing the value of SN to the sequence number field; the letters of the packet to the information field and the checksum (computed from the letters[3]) to the control field.
4. Copies (using pencil) the contents of the form to a transmission form, and passes it to the Physical layer group.
5. Waits for a response or for a time out notification. (The time out message is generated by the Physical layer group rather than by a real timer).
6. Increments the value of SN by 1 and goes to step 2, if a positive response frame (Ack) arrives. Otherwise (negative response or time out), goes to step 4 for retransmission.

III. The Physical layer group:

Upon reception of a frame from the Data-Link sender group, it throws the die (in such a way that the Data-Link group will not see the outcome) and acts as follows:

- i1. If the outcome is 1, it replaces one of the letters in the frame and passes the corrupted frame to the Data-Link receiver group.
- i2. If the outcome is 2, it destroys the frame and tells the Data-Link sender group that a time out has occurred.
- i3. Otherwise (3 to 6), it passes the correct frame to the Data-Link receiver group.

After giving the frame to the Data-Link receiver group, the Physical layer group waits for a response frame. When it gets the response, it throws the die, and acts as follows:

- i4. If the outcome is 1, it destroys the response frame and tells the Data-Link sender group that a time out has occurred.
- i5. If the outcome is 2, it tells the Data-Link sender group that a time out has occurred, but after a while passes the response on to it.
- i6. Otherwise (3 to 6), it passes the response frame to the Data-Link sender group.

IV. The Data-Link receiver group:

[2] These are translations of the (Hebrew) instructions that were given to the students.

[3] To compute the checksum, we used a Hebrew system (called Gimatria) of using alphabet as numerals. Any other (reasonable) function can serve for this purpose.

1. Writes on a piece of paper 'RN : 0'. The paper represents a Request (for frame) Number variable, RN, which is initialized to 0.
2. Waits for an arrival of a frame. When a frame arrives, it checks: i) if the checksum is correct, and ii) if the sequence number of the frame equals RN.
3. If the frame is OK, the group: i) passes the information field of the frame to the Network receiver group, ii) increments RN by 1, and iii) passes the Physical layer group an Ack response,. Otherwise, it passes the Physical layer group a Nak response.

V. The Network receiver group:

Constructs a message from the information pieces given to it by the Data-Link receiver group.

Step C: Reviewing the Game and Analyzing the Efficiency of the Protocol

The instructions for playing the game given above are (intentionally) incomplete, and may drive the protocol to a deadlock! If a frame is received successfully, but its Ack-response reaches the sender after time-out, then the sender retransmits the original frame (thinking the original frame, or its response, were lost) although the frame was properly received and accepted. This sequence of events is simulated in the game with instruction #i5 of the Physical layer group.

When attempting to explain this unfortunate sequence of events in a frontal way, the teachers found that most of the students could not understand how the problem was created, and why the protocol (as presented) could not handle it. Whereas, while playing the game, the students encountered the problem and could see its causes directly.

When the problem is encountered, the teacher stops the game and discusses the problem, and its solution, with the students. Following the experience gained in the game, most students do not have any difficulty in understanding the problem. The problem is then solved by the students with the teacher's assistance. In some classes students even solved the problem by themselves (numbering the response frames, instead of using unnumbered Ack and Nak responses).

After the problem is solved, the instructions of the game are corrected in accordance with the solution, and the game resumes for a while - this time without problems.

When the teacher feels that the protocol is fully understood, she stops the game and moves to discussing the efficiency of the protocol. The link utilization of this protocol is determined by the frame *transmission time* and the *propagation delay*. These two abstract terms are illustrated easily using the game: frame transmission time is the time it takes for the Data-Link sender group to compute the checksum and fill in the form representing the frame; while the propagation delay is the time it takes for the Physical layer group to travel between the two Data-Link groups.

## Discussion

Students, teachers and us, developers of the program, are convinced that the game is a useful tool for teaching communication protocols in a very vivid and concrete way. After each experimental operation of the game we conducted a final discussion and asked the students to evaluate the game. The students' reactions were enthusiastic, and affirmed the impression we had while watching them play. It seems that the random nature of the game contributes a lot to the interest the students exhibited. The acts of throwing the die and seeing what the outcome was, were peak moments that attracted the attention of the players; the Data-Link students wanted to know what the outcome was, but they had to figure it out indirectly through the actions of the Physical layer group.

This specific experience of uncertainty, was the strongest contribution of the game for understanding the nature of the communication protocol. As already mentioned, frontal explanations of concurrent and distributed processes are difficult to grasp. This is true even if one uses conventional illustration means such as figures and slides. In frontal explanations, the learner is presented with a sequence of events that contains complete information about a situation that the protocol may run into. But since there are several agents in several sites, and uncertainty is involved, each agent does not know what is happening in other sites. For many students, it is hard to keep track of what is known to each agent, and to see how agent actions can be determined from only partial knowledge. When the student is actually playing the role of an agent, and experiences its point of view, there is no need for mental visualization: the problems, along with their solutions, become clear.

In order to verify our impression quantitatively, we compared results of a written test given both to students who learnt the protocol in the standard way (no game), and to students who learnt the protocol via playing the game. This comparison was done in two of the classes in which we applied the game (about 60 students). The average grade of the students who played the game was 87, while the average grade of the other students was 68.

Apart from the main advantage of the game - its illustrative characteristic - the game had several additional advantages. It fostered *group cooperation* in which the majority of the students are actively involved in the learning process. Collaborative learning and student involvement are desired activities that are, unfortunately, too rare in traditional learning settings [8]. The game also created a process of *discovery learning*; this kind of learning is known to improve self learning and problem solving abilities [1].

An example for the involvement and motivation fostered by the game is an unanticipated effect that was created in some classes when students from the Physical layer group, that had to corrupt a frame, replaced two letters instead of one in such a way that the checksum of the frame remained unchanged. Thus they created an undetectable error. This unplanned action created opportunity for discussing the limitation of the error detection methods and the probability that errors in a frame will go undetected.

In summary, there were several assets to the game introduced:

- it served as a tool for understanding the difficulties in distributed and concurrent processing
- it simplified and clarified the non trivial stop & wait protocol
- it made the abstract terms needed to discuss the efficiency of the protocol concrete
- it fostered discovery and collaborative learning.

We believe that the success described above should encourage others to teach non trivial algorithms through game simulation.

## References

[1] Baldwin, D. "Discovery Learning in Computer Science", *Proc of the 1996 SIGCSE Technical Symposium on Computer Science Education,* Feb 1996, pp 222-226.

[2] Brown, M. H., "Perspectives on Algorithm Animation", *proc. ACM SIGCHI, '88 Conf. on Human Factors in Computing Systems,* April 1988, pp. 33-34.

[3] Chung, C. M., Mak, S. Y., Suen, Y. M. and Sze P. "Game-Display Board Activities for Science Teaching", *Journal of Science Education and Technology,* Vol. 5, No. 2, 1996.

[4] Felder, Richard M. "Reaching the Second Tier - Learning and Teaching Styles in College Science Education", *Journal of College Science Teaching,* pg. 286-290, 23(5), March/April 1993.

[5] Ginat, D. "Loop Invariants and Mathematical Games", *Proc of the 1995 SIGCSE Technical Symposium on Computer Science Education,* 1995.

[6] Harvey, B. "*Computer Science Logo Style, Volume 1: Intermadiate Programming*", MIT press 1985.

[7] Levy, D. "Metaphors in the Teaching of Computer Science in High-School", *master thesis,* Technion, Israel, 1991.

[8] Ramsey, P., Rada, R. and Acquah, S. "Collaborative Learning for Computer Science Education" *Journal of Computers in Mathematics and Science Teaching* 13(4), 377-389, 1994.

[9] Rasala, R., Proulx, V. K., and Fell, H. J. "From Animation to Analysis in Introductory Computer Science", *Proc of the 1994 SIGCSE Technical Symposium on Computer Science Education,* 1994.

[10] Shifroni, E., and Zilberstein, I. "The OSI Model as a Methodological Framework for a Course in Computer Communication", *the 2nd Jerusalem International Science & Technology Education Conference (JISTEC)* 1996.

[11] Tanenbaum A. "*Computer Networks*", 3rd edition, Prentice-Hall 1996.