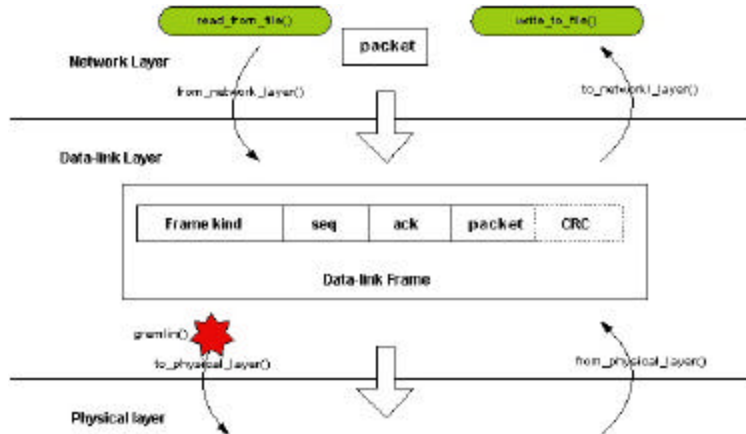


**COMPUTER COMMUNICATION NETWORKS**  
*ECE 333- Spring 2004, Computer Project 3*  
**Data Link Layer\***

**Objective:** Simulating the data link transmission protocols.

**Procedure:** You may work in teams of **two** for this project. Hand in a single report for the team. Make full use of the sample data-link layer simulator posted on the lab page. You will have to modify this code to include the CRC and gremlin functions at the data-link layer. The following diagram explains the data-flow model in more detail.



**Background:** The following Data Link Layer Protocols should be simulated:

- Unrestricted Simplex Protocol
- Simplex Stop-and-Wait Protocol
- Positive Acknowledgement with Retransmission Protocol
- Sliding Window Protocol
- Sliding Window Protocol Using Go -Back-N
- Sliding Window Protocol Using Selective Repeat

**Procedure:**

1. Write a sample program in C to create an ASCII data file containing at least 1000 alphanumeric characters.
2. Understand the data-link simulator that has been posted on the lab-page. This simulator can also be downloaded from text - book's supporting web-site (<http://authors.phptr.com/tanenbaumcn4/>). Instructions for downloading, untarring and compiling this simulator will be posted on the lab-page (<http://mia.ece.uic.edu/~papers/ece333/spring04/>).
3. Modify the transmitter function for each data-link protocol (for example, p2.c for stop-and-wait protocol). This transmitter function must do the following:
  - Reads data from the input data file created in (1).
  - Creates data-link frames. Each frame should only contain 8 bytes of input data.
  - Calculates the checksum (when necessary). Use the CRC-CCITT for calculating the checksum. An example implementation has been posted on the lab-page.
  - Applies the gremlin function to corrupt the packet, if necessary.
  - Sends the data-frame to the receiver.
4. Program a receiver function for each data-link protocol.
  - Checks the checksum (when necessary) and properly handles errors.
  - Returns an acknowledgement or negative acknowledgement (when necessary).
  - Writes the frame to the output data file.

5. Plot the number of correct frames received by the receiver after transmission is complete as a function of the probability that a frame will be damaged (for each data link protocol).
6. Plot the time required for transmission to be complete as a function of the probability that a frame will be damaged (for each data link protocol).
7. Determine the noise immunity (for each data link protocol).
8. Determine the time requirements (for each data link protocol).

**Note:** The gremlin is a function that is used as a virtual communication system. It will be used to simulate transmission of damaged frames with some probability due to noisy communications networks. For simplicity, however, all transmitted frames are assumed to arrive at the receiver. That is, data frames may be damaged but not lost.

**Comment:** The time elapsed is determined in one-way path units (i.e., time required for transmission from the source to destination, or vice versa). Consider a single time-unit to correspond to 250 milliseconds.

**Suggested Software:** C/C++.

**References:**

[1] F.S. Grodzinsky, Ed., *Networking and Data Communications Laboratory Manual*, Prentice-Hall, 1999.

\* **Note:** This experiment has been developed by D. Schonfeld, Electrical and Computer Engineering Department, University of Illinois at Chicago, Chicago, Illinois. Some of the concepts have been inspired by P. Sanderson, Computer Science Department, Southwest Missouri State University, Springfield, Missouri [1].

\*Modified by Shashank Khanvilkar.