

CSE21/31NET & CSE41FNT DATA COMMUNICATIONS AND NETWORKS

ASSIGNMENT 1

Introduction

The aim of this assignment is to write a program, **dlsim**, that simulates a simple “stop and wait” data link protocol (also known as an alternating bit protocol). This is an **individual assignment** and introduces the concepts involved in simulating a simple data link level protocol. (Note that as this is a simple protocol we can simulate its behavior by repeated calls, in order, to 3 **functions** while there is more data to send.)

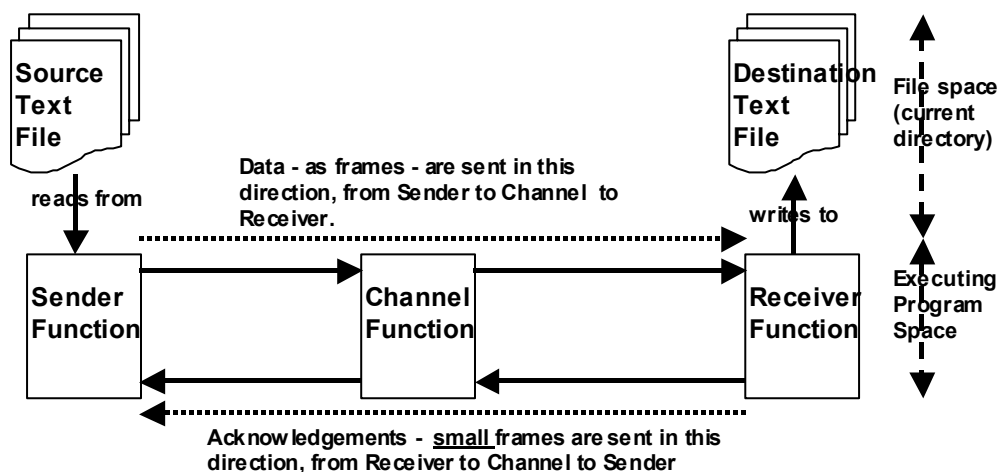
Objectives

This assignment is designed to achieve the following objectives:

- Introduce a simple data link protocol,
- Understand the basic concepts involved,
- To be able to design a suitable simulator, then map/translate the basic concepts into a suitable programming language,
- To implement a simulator that performs the tasks required,
- To perform some basic tests,
- Write an objective report on the assignment/project, and
- Demonstrate a working product (simulator) and be able to identify and discuss salient features/limitations.

Outline

Diagrammatically the program simulates the following:



Basic operation of the program

The 3 functions should be designed to perform the following tasks:

1. **Sender:** The sender function is to construct a frame to be transmitted, including a data field, a checksum field and a frame sequence number. The data field is to contain 10 characters and is to be read in from an

existing text file that contains at least a page of text, in your current directory. (The file to be used as the source text file is determined each time you run the program.) A **CRC** calculation is to be undertaken for each new frame and the remainder is to be placed in the checksum field of the frame. The frame sequence number field is to toggle between 0 and 1 (hence AB protocol), depending upon the frame being sent. The sender is to be able to send each frame to the channel and then receive a small acknowledgement frame back from the channel and, depending upon the acknowledgement, perform the following tasks:

- a. **ACK** (positive acknowledgement): construct a new frame (get more text from the source file) as above and send it to the channel, or
 - b. **NACK** (negative acknowledgement): re-send the previous frame.
2. **Channel:** The channel is designed to simulate the physical layer behavior between the sender and receiver. Depending on the particular cases it is simulating, the channel is required to be able to:
 - a. (**CSE21/31NET & CSE41FNT**): Forward to the Receiver the frame without change,
 - b. (**CSE21/31NET & CSE41FNT**): Introduce an error into the frame by inverting a bit before sending it onto the Receiver,
 - c. (**CSE41FNT only**): Drop the frame (i.e. just not forward the frame onto the Receiver), and
 - d. (**CSE21/31NET & CSE41FNT**): Pass back from the Receiver a small acknowledgement frame **without altering it**.
3. **Receiver:** The receiver is designed to accept a frame from the channel and checks the frame to see if it is in error by computing its own checksum (using exactly the same CRC polynomial (CRC function) used by the sender) and comparing it with the checksum field that arrived with the frame. If they are the same then the frame is OK and the receiver immediately writes the data (which is text from the source file) to the destination file and constructs a small acknowledgement frame containing an **ACK** and sequence number **of the next frame to be sent**. If the frame was in error, then an **NACK** is sent back in the acknowledgement frame along with the **sequence number of the frame to be re-sent**.

CRC Polynomial

The actual polynomial used in your CRC function should be a prime number. You may use a well-known polynomial (See Stallings, p183) such as CRC-16 (recommended).

Program execution

The program is to be executed by the following command line entry:
dlsim "source-filename" "destination-filename" "message-filename"

Where:

- **dlsim** is the name of the executable (default is **a.out**)
- **"source-filename"** is replaced by the actual name of the file containing the source text.

- “**destination-filename**” is replaced by the name of the actual file (which may not yet exist) in which the receiver is to place the text it receives from the sender.
- “**message-filename**” is replaced by the name of the actual file in which a recording of the sequence of events as undertaken by each function is performed. Typically this file should appear as a simple listing of events as they occur in your program – the exact detail is left to you but at least something like:
sender: frame # sent
channel: frame # received , not corrupted
receiver: frame # received, OK, ACK returned
channel: ack_frame received – passed to receiver
sender: ACK received, new frame # sent
channel: frame # received, corrupted
receiver: frame # received, corrupted, NACK returned
channel ack_frame received – passed to sender
sender: NACK received, re-send frame #
(etc for the duration of the file transfer)

The **channel function** is to have the following corrupting and dropping frequencies:

- 10% of the time that a frame is received from the sender it is to be corrupted by “flipping” a bit (**CSE21/31NET & CSE41FNT**),
- 1% of the time a frame is received from the sender it is to be dropped i.e. “effectively” not sent or forwarded to the receiver (**CSE41FNT only**), and
- Additionally the small acknowledgement frames are simply passed back from the Receiver to the Sender and **never corrupted**.

Programming Language

You can implement this program in C, C++ or JAVA. **The final, submitted version must be compileable and runnable on latcs(x) machines (running Linux operating system) in the Computer Science department. A program that fails to compile or run on a latcs(x) machine will automatically result in a fail grade being awarded.**

Hand in Requirements

You are to hand in a **physical (paper) copy** of the following along with a **firmly attached and signed** certificate of authorship of your submitted work. **A failure to submit any of the following will automatically result in a fail grade being awarded.**

1. Program source listing,
2. A listing of your source file, destination file and message file which you may annotate by hand for comments or explanation,
3. A report outlining the assignment including comments and/or discussion on:
 - a. General description of the program/assignment (e.g. protocol involved, any other ways to simulate an AB protocol?) and a

- general description of the operation of your program (command line entries, any limitations, performance, etc),
- b. The algorithms used,
- c. Data structures used/implemented,
- d. Analysis of results (test cases tried, is it slow – why? are there any differences between source and destination files, any other observations/comments you wish to make).
- 4. A script session showing your program running, and
- 5. **An electronic submission of your program listing (e.g. *dlsim.c*) to a submit account for Data Communications and Networks – (to be confirmed later but normally as CSE21NET, CSE31NET and CSE41FNT). (Note that several programs are used to test for any occurrences of plagiarism. Each program will be electronically tested against all other submitted programs.)**

Marking Outline

Marks will be allocated according to:

- | | |
|---|------|
| 1. Program coding and operational correctness | (35) |
| 2. Algorithms and data structures | (15) |
| 3. Internal program documentation | (10) |
| 4. Written assignment report | (30) |
| 5. Modularity, flexibility, etc | (10) |

Plagiarism

Students are reminded of the penalties for plagiarism is outlined in the student handbook for 2004 (Section 4.7). All electronic copies will be scanned electronically for any occurrences of plagiarism and any detected instances will be viewed seriously and procedures instigated as outlined in the student handbook (2004). In relation to the general procedures for submitting and subsequent processing of assignments students are directed to the following Sections of the student handbook 4.6, 4.8, 4.9, and 4.10,

Due Submission Date

The due submission date is **Wednesday, 19 May 2004 at 9.00 AM**. An assignment submission box will be clearly marked for assignment submission as “**CSE21/31NET&CSE41FNT**”, on the ground floor of the Beth Gleeson building. The submission date and time is exactly the same for both electronic and “hard” (paper) submissions.

Late Penalties

Students are reminded of the prescribed penalties for a late assignment submission (Section 4.6 of the student handbook, 2004) where, “A penalty of 5% per day will be imposed on **all late** assignments up to **4 days late**. An assignment submitted more than 4 working days after the due date **will not be accepted**. Weekends are not included in the late period”.

Return of assignment and function test/demonstration

There is to be a function test/demonstration of your program before a mark/grade will be given. The date/time for the function tests will be promulgated shortly after Wed, 19 May 2004. (In general the function tests are normally scheduled to be about 2 weeks after the assignment submission date to allow for initial marking, plagiarism testing and availability of labs etc.) At the completion of the function test your marked assignment will be returned to you. Any students failing to demonstrate their program will **automatically be graded a fail**. If any student cannot attend the demonstration at the promulgated times, they **must arrange a suitable alternate time with the Tutor-in-Charge before they fail to turn up**, otherwise a fail grade will be awarded.