University of Waterloo

Faculty of Engineering *Department of Electrical and Computer Engineering*

ECE 428 – Computer Networks and Security
Project 2: UDP-based TCP Communication
Spring 2010
Instructor: *Mahesh V. Tripunitara (tripunit@uwaterloo.ca)*
TA: *Ahmad R. Dhaini (adhaini@uwaterloo.ca)*

1. Objective

The purpose of this project is to emulate a TCP connection using UDP sockets. More specifically, you are required to use a non-reliable, connectionless and packet-oriented connection to provide a reliable, connection and stream-oriented one.
The project will also give you some insights on the understanding of TCP in terms of connection establishment and stream sustainability.

2. Description

Given that you have learned in the first project how to setup a communication between a client and a server using both TCP and UDP connections, you are now required to demonstrate how TCP ensures reliability using the non-reliable UDP protocol.

In this project, you have to implement, using either C/C++ or JAVA programming language, an API (or skeleton file) that emulates the usage of TCP on top of a given UDP-based API.

In addition, you are required to use the implemented API to collect some statistics related to the performance of your implemented protocol.

Namely, the project is divided into the following two folds:

A. API Implementation

Using the given UDP API, `T_DatagramSocket.c (java)`, you are required to implement `S_StreamSocket.c (java)`, to provide a TCP-like communication. A TCP-like communication means that a connection should be first established between a *client* and a *server*, and the data communication should be reliable and

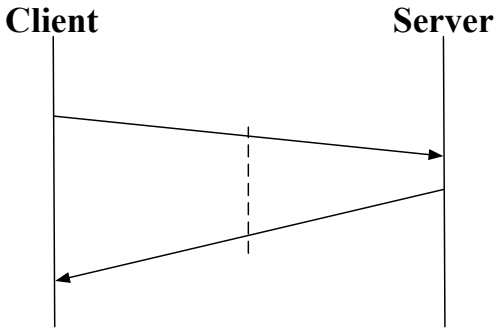
stream-oriented. More specifically, your program should be able to recover from any data corruption or packet drop when invoking the `T_sendto` function implemented in the given UDP API.

Note: You should use the `T_DatagramSocket` methods almost entirely except for some functions, such as `ntohs`(), that are not included.

B. Collecting and Reporting Statistics

To make you familiar with data and simulation analysis, you are required to do the following:

a) Pictorially show how connection-setup (with all the required labels) works in your submission for the project. The illustration may look like this:



b) Draw your design of the state machine for a connection.

c) Presumably you have written a client & server that uses the `S_StreamSocket` interface to test your submission, perform the following (in the same order):

- **i.** Run a server that accepts a connection, and a client that establishes a connection.
- **ii.** Send the following amounts of data over that connection from the client to the server. (1) 1k bytes, (2) 5k, (3) 20k, (4) 75k, (5) 250k.
- **iii.** Measure the (real) time it takes for the data to be fully received at the server.
- **iv.** Draw a graph (the horizontal axis is the data size from above, the vertical axis is time).
- **v.** Reason about why your graph looks like what it looks like based on how we have implemented `T_sendto` ().

3. Things to Consider

a. Testing your program

To test your API, you have to write your own client and server. These shall not be delivered with your project submission. The input could be taken from a file or via the command line.

b. Expected Output

The program can be as simple as sending one message via your implemented API, or transferring a big chunk of data back and forth between a client and a server. The data should be delivered correctly at both ends (i.e., packet corruption and/or drop should be recovered).

4. To Submit

Each project should be submitted as a gzipped tar file through UW-ACE. The naming convention should be as follows:

g*N*_p*M*.tar.gz

N stands for the group number and *M* stands for the project number. For example, group 20 shall submit project 1 using:

g10_p1.tar.gz

If you make a mistake during a submission, you may simply resubmit, and the original submission will be ignored. If you feel that the wrong file was marked (UW-ACE requires to manually select the most recent project), please contact the TA.
Make sure to use lower case for "p" and "g"; otherwise your submission will be ignored.

Your gzipped file should contain the following files:

- `S_StreamSocket.java – or – S_StreamSocket.c`
- g*N*_p*M*_report.pdf

Note: Only *pdf* formats are acceptable for the report. Also, if you decide to submit any additional files, make sure to submit a `makefile` that will produce `S_StreamSocket.o(.class)` as main target. Failing to do so might result in deducting marks from your grade.

5. Grading Policy

Grading will be done as follows:

Total is 100 points. You will receive:

- 20 % if your program compiles, but does not work at all.

- 0 points if your program does not compile.
- 0 points for identical (or very similar) programs.
- 60 % if your program passes the basic test case, but is badly written[1].
- 65 % if your program passes the basic test case and is well written[2].
- 95 % if your program passes all the test cases, but is badly written.
- 100 % if your program passes all the test cases and is well written.

Late submissions will not be accepted.

It's better to have something working, even with little functionality, than a program that crashes (or doesn't compile). It is expected that you submit your own program; otherwise you will get a 0 for the project.

6. Deadline

Monday June 21, 2010 before 11:55 PM.

[1] Badly written code means that the code is neither commented nor follows the *programming style*. A good programming style guideline can be found at: http://en.wikipedia.org/wiki/Programming_style

[2] Reviewing your grade because of programming style is not allowed.