INF-2301: Assignment 3: Reliable Data Transport in network communication

Submission Deadline: Monday, 29 October, 2018 23:59:59 (CEST)

1 Introduction

In this assignment, you will design a *reliable transport layer* in an OSI-network stack. The purpose of your transport layer will be to mask various errors occurring in the lower layer of the network stack, in particular package loss and reordering. Your transport layer should provide an error free and continuous stream of data to the above layers, matching what the sender writes to the network.

There are various ways to provide reliable data transfer, each with its own drawbacks and benefits. In this assignment, you are encouraged to explore and implement a solution based on either *the Go-Back-N* Protocol, the *Alternating Bit* Protocol, or a *Selective Repeat* solution.

Most real-world reliable transport protocols use variations or extensions of the *Go-Back-N-Protocol*. Even TCP normally operates in Go-Back-N mode at its core, though it also includes additional functionality. An implementation of the Go-Back-N-Protocol might be a good choice to achieve reliable data transfer. The *Alternating Bit-Protocol (ABP)* can be thought of as a simplified version of the Go-Back-N-Protocol, and is a possible candidate for solving this assignment. However, using this protocol, you might encounter some complications due to its inherent limitations, and implementation complexity might not differ that much from a Go-Back-N solution.

A more complete solution to this assignment would involve a protocol that use Selective-Repeat. This can be done as an extension to an already implemented Go-Back-N solution. There are several convenience functions available in the pre-code to help achieve selective-repeat functionality as the timer objects only allow for a single context parameter. Note that although TCP has Selective-Repeat extensions, its default operating mode, which builds on a classic Go-Back-N solution, is so highly optimized that selective-repeat is rarely activated.

For a visual representation of what this assignment entails, you can visit the following website: http://www.ccs-labs.org/teaching/rn/animations/gbn_sr/

2 Pre-code

To simplify development and testing of your transport layer, this assignment includes a network simulator in the pre code. As such, you do not need to deal with the complexity of real network connections or the OSDI stack in your OS. You are only required to extend the transport layer in the simulator with your own transport layer implementation.

The network simulator is of some complexity, and is implemented over source code files. You are encouraged to explore all of this, but to solve this assignment you only need to focus on the transport layer implementation. The pre-code contains self-documenting comments from which we have generated documentation pages that you can view in your web-browser (the documentation pages are generated in the HTML document format). In the documentation pages, you will find extensive descriptions on all functionality of the network simulator as well as a general overview on what the functions you are to implement should do. It is therefore highly recommended to have a closer look at the documentation pages. Some of you might even find the “Where should I start?” section on the main page helpful in the beginning. Read the documentation by opening “index.html” (located in the doc directory) in a browser.

3 Requirements

For this assignment, we expect code and documentation of your reliable transport layer. Documentation should be in the form of commented code and a report. You should describe carefully what your transport layer implementation does when it receives data from both the application layer above and the network layer below. You should also document and discuss the types of irregularities a transport package might encounter during transmission and your how implementation deal with those. Note that there is at least one case in which no regular Go-Back-N implementation (not even TCP) can protect against an invalid transmission. In an ABP implementation, this case is very common. Why? In addition, what do we do to address this issue?

If your implementation is only a simple Alternating-Bit implementation, you are REQUIRED to put the Alternating-Bit-Protocol into the context of Go-Back-N by explaining the similarities and limitations in relation to a Go-Back-N implementation.

4 Hand-In Instructions

Submit your hand-in via Canvas before the specified deadline. Your hand-in should consist of a separate pdf-file, as well as a SINGLE archive file such as a Zip-compressed Archive of your choice(.7z, .tar, .tar.gz, .tar.bz, .tgz, .tbz, etc.) or a WinRAR-archive (.rar). The name of the hand-in file should be in the following format: inf2301-2018.2-1-abc123.tar.gz, where abc123 is your student name.

Inside the archive folder, there should only be a single folder with the same name as the archive filename. This step is to ensure that multiple hand-ins can be extracted safely within the same folder by the grading TA.

Underneath that root folder, there should be a subfolder called “src” containing the source code. The hand-in should also include a README file in the root folder where you give source-code-specific instruction on how to compile and execute your source code.
Your report is handed in separately in Canvas as a pdf-file.

Reasonable use of external non-standard libraries is okay, but only if their use is properly documented in the hand-in and you give instructions on how to obtain them.