Inf-2301: Assignment 3

Submission deadline: 23:59 Friday October 9 2015

1 RTFM

For the Inf-2301 assignments, we are not going to provide you with complete information about protocols and the tools you need to use in order to answer all questions and write your code. You will be expected to nd and read the documentation that is widely available on the Internet or on the machines in the student lab (in the form of manual pages for example). Your fellow students can also be a great source of general information. When the information you have collected is not clear enough or you get stuck in your search for relevant documentation, you can always ask for assistance from the Inf-2301 staff. Until then, the following assistance to RTFM may be of help.

2 Manual pages

If you have never used the **man** command before, it is time to start now. The **man** command on Unix-like systems will give you a manual page for most commands and tools available on the system, for example: **man pwd** gives you an explanation of the **pwd** command. Another source of information (sometimes better than manual pages on GNU/Linux systems) are the so-called info pages. Try for example: **info pwd** for the info page on the **pwd** command. Reading manual pages effectively is something you have to learn, and spending some time 'getting used to them' typically pays itself back later. Reading the following manual pages are a good start: *man*, *apropos*, *whatis*, *whereis*, *locate* and the introduction pages for each manual section.

2.1 Internet Standards

Another important source of information you will need to consult regularly are the actual Internet standards produced by the IETF (The Internet Engineering Task Force) and other organizations. Each distinct version of an Internet standards-related specication is published as part of the "Request for Comments" (RFC) document series. This archival series is the offcial publication channel for Internet standards documents and other publications of the Internet Architecture Board (IAB) and the Internet Engineering Steering Group (IESG), and Internet community. RFCs can be obtained from a number of Internet hosts using anonymous FTP, the World Wide Web, and other Internet document-retrieval systems. If you are interested in the Internet standards process itself or just want a better understanding of the terminology used in them, then RFC 2026[1] may be of interest. Also, the IETF newcomer's presentation[2] or the "Tao of the IETF"[3] give a good idea of the standardization process and the organizations involved.

[1] Search for RFC at http://www.ietf.org/rfc.html (just type in the RFC number)
[2] http://www.ietf.org/proceedings/07dec/slides/newcomer-0/sld1.htm
[3] http://www.ietf.org/tao.html

2.2 Other Sources of Information

Often overlooked nowadays (because you cannot click on them?), but extremely valuable sources of information are libraries. A library is also something you need to learn how to use, so it never hurts to 'just' spend some time in one to get familiar with what treasures can be found there. If you prefer to click on things and stare at computer screens you can turn to a digital library. An excellent example is the digital library[4] of ACM (Association for Computing Machinery). At the welcome page of the ACM Digital Library you will see "University of Tromso" when accessing this resource from computers on campus. Another good source is IEEE Xplore[5]. At the welcome page of IEEE Xplore you should see the message "Access provided by: UNIVERSITY OF TROMSE" when accessing this resource from computers on campus. Be aware that both digital libraries will give you only limited access if your computer is not recognized as a campus computer. This is services that our university library pays for, and access is granted based on IP-addresses.

3 Introduction

The reason that reliable transfer protocols are important today is to ensure that *all* data arrives, and that it arrives in order and is not corrupted. The most common protocol is the Transmission Control Protocol (TCP), which has a number of features such as congestion control and flow control, but we will focus on the reliability part. An alternative to reliable protocols is the User Datagram Protocol (UDP), which gives no guarantees about reliability.

In this assignment we will implement and analyze a simple reliable data transfer protocol. More precisely, you will be writing the sending and receiving transport-level code of a simple reliable data transfer protocol. In reliable data transfer the data traffic across a network is mostly comprised of two types of packets: Data packets and Acknowledgement packets. The data packets contain information about the packet itself (such as Sequence Number, Checksums, Data size etc.) as well as data, while ACK packets are only meant to acknowledge the last received data packet and does not contain any data. Every time a packet is not Acknowledged, or the acknowledge does not arrive in time, or the packet arrives out of order, the previous packet is resent. You will need to read up on network packets and how they work in order to understand this assignment.

Part A and Part B are two different reliable transfer protocols and it is optional to do both part A and part B. You can choose to do either part A or part B. However, we recommend that you do both parts.

3.1 Pre-Code

For this assignment you will be provided with some pre-code that simulates the network environment. You are welcome to look at the files in the pre-code, but you are not required to understand it, what is important is the Transport Layer part where you need to implement the missing functionality. The code provides simulation a number of things such as network delay, packet corruption, packet loss, speed of data etc. You should use different settings to test your implementation and write and discuss the results in your report.

[4] http://portal.acm.org/dl.cfm
[5] http://ieeexplore.ieee.org/

4 Part A: Alternating-Bit-Protocol

This assignment is about a reliable data transfer protocol. You should implement a Alternating-Bit-Protocol (ABP)[6] on top of a simulated network environment.
The ABP, also known as the Stop-and-wait ARQ (Automatic Repeat Request), operates by sending one network packet at a time, then waiting for that packet to be acknowledged or ACKed. The sequence number of the packets alternates between 0 and 1, hence the name alternating bit. If a packet is received out of order, which means the recipient receives two consecutive packets with sequence number 1, the new packet must be discarded. To signal the sender that a packet was lost, you can send the ACK for the last received packet. For more details you can follow the link in the footnote.

4.1 Implementation Details

5 Part B: Go-Back-N

This part is about another reliable data transfer protocol. You should implement a Go-Back-N on top of a simulated network environment. The Go-Back-N (GBN)[7] protocol is a type of sliding window protocol. This means that there are a series of packets being sent before waiting for an Acknowledge of all packets or the last one of the window. The recipient will acknowledge the the highest current packet that has been received and was in order and not corrupted.

If packets are lost, you should still acknowledge the last successfully received packet to tell the sender where he should start resending packets. The so called "sliding window" starts at a *base* and ends at the *base + n-1* which tells the protocol which packets should be sent and how many at a time. When you receive an acknowledgement, the sequence number of the acknowledgement becomes the new *base*. You also need to chose a window size for the sliding window, and you can experiment with different sizes. For more details you can follow the link in the footnote.

5.1 Implementation Details

6 Report

For part A describe the design and implementation of the programs you made. Do not simply copy a lot of content from external sources like RFCs, refer to them instead. Do not include the source code in the text (it should be submitted as separate files). However, the source code should be readable enough, both in structure and the way it is commented, for another programmer to verify its correctness. In other words, your report with source code needs to convince us that your programs work, not execution of the actual programs! Include also a description on how you tested your programs.

For part B you should produce an analytic text tied to your description of part A. The text should not assume that the reader know the detailed content of your text books, and it should refer to external sources used.

Include a 'discussion section' in which you note your observations on this assignment, the tools, protocols and libraries you studied, and the documentation you consulted. Maybe you observed something interesting (or just surprising to you). If so, write about it in your report.

Describe what aspects of the solution that caused you implementation headaches, if any. Also report any particular assistance you got from course staff or your fellow students that has been crucial to the fulfillment of the assignment. If an RFC or document was hard to read/interpret

[6] `http://www.erg.abdn.ac.uk/users/gorry/eg3567/arq-pages/saw.html`
[7] `http://www.erg.abdn.ac.uk/users/gorry/eg3567/arq-pages/gbn.html`

for you, make a note of it. You also need to document your sources, this means citations in your report. If you do not know how to cite in technical reports you can either speak with one of the TAs or read up on citation. Wikipedia[8] has a good explanation of citations (since most of Wikipedia is Citations), although it is not a good source to cite itself. You are strongly advised to write documentation/notes along the way.

7 Deliverables

This assignment must be fulfilled by every individual student (no group delivery). For this assignment you need to use C programming language due to the pre-code being implemented in C.

Please follow these guidelines for the delivery of your result. Create a directory named after your UiT user name (e.g. abc012). This directory should contain all your deliverables. Assignment documentation and report can be composed in any program, but must be delivered either in adobe acrobat format (pdf) or postscript (ps).

For source code, create a sub directory called `src`. This directory should contain a README file in plain ASCII text, describing how to build (compile) your project, install it (if needed) and how to run it (arguments, etc).

The entire directory tree should be archived in a tar/gzip or zip file. This file should be named as the assignment directory (with the appropriate `.tar.gz`, `tgz` or `.zip` suffix). Finally use Fronter to upload this file to the "Inf-2301 assignment 3" hand-in folder in the Inf-2301 room.

[8] `https://en.wikipedia.org/wiki/Citation`