**SCHOOL OF INFORMATION TECHNLOGY AND SCIENTFIC COMPUTING**

**ADDIS ABABA INSTITUTE OF TECHNOLOGY, Addis Ababa UNIVERSITY**

**Distributed Systems Programming – ITSE-5243**

**Deadline: January 30 2022**

**Programming Assignment 1: BitTorrent Client Implementation**

**Total Weight: 25%**

# Overview

In this assignment, you and your lab partner will develop a basic BitTorrent client that can, at minimal, exchange a file between multiple peers, and at best, function as a full feature BitTorrent Client. The programming in this assignment will be in Golang requiring standard sockets, and thus you will not need to have root access. You should be able to develop your code on any CS lab machine or your personal machine.

This assignment has been designed to demonstrate the use of two fundamental technologies that have been discussed during the lectures:

* *Sockets*
* *Goroutine*
* *Channels/Messages*

Hence, the assignment must make an **EXPLICIT** use of the above. By explicit, we mean that in your application, sockets, goroutine, channels must be the lowest level of abstraction for network communication and concurrency.

# BitTorrent Protocol

BitTorrent is a peer-to-peer file sharing protocol develop by Bram Cohen. It was developed in response to centralized file sharing services, like Napster, where centralized infrastructure was necessary to enable downloaders to connect with uploaders. Additionally, centralized services did not enable for multiple file uploaders to send data to multiple file downloaders.

BitTorrent is designed to address both of these shortcomings by enabling file downloaders and file hosters to interact directly in a peer-to-peer fashion; that is, BitTorrent participants interact using a peer to-peer protocol where each participant acts as both a server and client. Perhaps BitTorrent must important feature is that it enables a large number of downloaders to collaborate in downloading a file from a much smaller number of file uploaders. BitTorrent also has built in mechanisms to ensure fairness which encourages participants to share previously downloaded data as uploaders, and if they don’t, a downloader is punished by limiting download speeds.

In this assignment, you will implement a core subset of the BitTorrent protocol by implementing a BitTorrent client. Below, I outline the relevant portion of the BitTorrent protocol, for more detailed descriptions, you should read Bram Cohen original specification here: http://www.bittorrent.org/beps/bep\_ 0003.html. Unless otherwise stated in this document, you are expected to implement your client to meet the specification outlined by Cohen

# Grading

Grading Breakdown Below is general grading breakdown that will be used for this assignment. Note that other issues will be considered, such as programming style and logic errors

* **1-Seeder and 1-Leecher (16%) :** Build a client that can implement the standard client protocol such that a single seeder can transfer a file to single leacher.
* **1-Seeder and 1-Leecher with Restarts (18%):** Build a client that can implement the standard client protocol such that a single seed can transfer a file to a single leecher, and the leecher can restart from a partial download.
* **1-Seeder and 1-Leecher with Restarts (20%):** Build a client that can implement the standard client protocol such that a single seed can transfer a file to a single leecher, and the leecher can restart from a partial download.
* **1-Seeder and 1-Leecher with Restarts (22%):** Build a client that can implement the standard client protocol such that a single seed can transfer a file to a single leecher, and the leecher can restart from a partial download.
* **1-seeder and N-Leechers with Restarts(24%):** Above, but clients can fail and restart with partial torrent downloads
* **1-seeder and N-peers (25%):** Build a client that can support a single seeder and multiple peers who also exchange with each other

The assignment should be implemented in **Golang**.

# Submission

You need to submit the following in class:

* Your source files in a .ZIP or .TAR archive *only*.

# Demonstration Schedule and Venue

You are required to provide a demonstration of the working application and will have the opportunity to discuss with the tutors the design and implementation choices made during the demo.

You are free to develop your system where you are more comfortable (at home, on one pc, on your laptop, in the labs...) but keep in mind that the assignment is meant to be a distributed system that works on at least two different machines in order to separate the client from the server.

**We will announce the demo date, time, and venue closer to the due date**. Each tutor will hold 2 demo sessions and you will be required to showcase your system in one of the sessions held by the tutor of the workshop in which you are enrolled.

If you need any clarification on the assignment, kindly ask questions during the tutorials or in the LMS forum, so that all students benefit from it.

**Note: You need to submit a printed copy of your report during the demonstration.**

# Penalties for late submissions of assignments

Assignments submitted late will be penalized in the following way:

* 1 day late: -1 mark
* 2 days late: -3 marks (-1 - 2)
* 3 days late: -6 marks (-3 - 3)  4 days late: -10 marks (-6 - 4)  etc.