Project 3

CS 381 - Intro to Computer Networks

Taylor Atkinson, Chris Goulet

### Introduction

This paper goes over how to run and use the Go-Gremlin project that was created by Christopher Goulet and Taylor Atkinson. It will also include some problems we encountered while doing this project and the resources we used. This project has only been tested on Ubuntu 15.10 (Wily Werewolf) with the Mate Desktop environment.

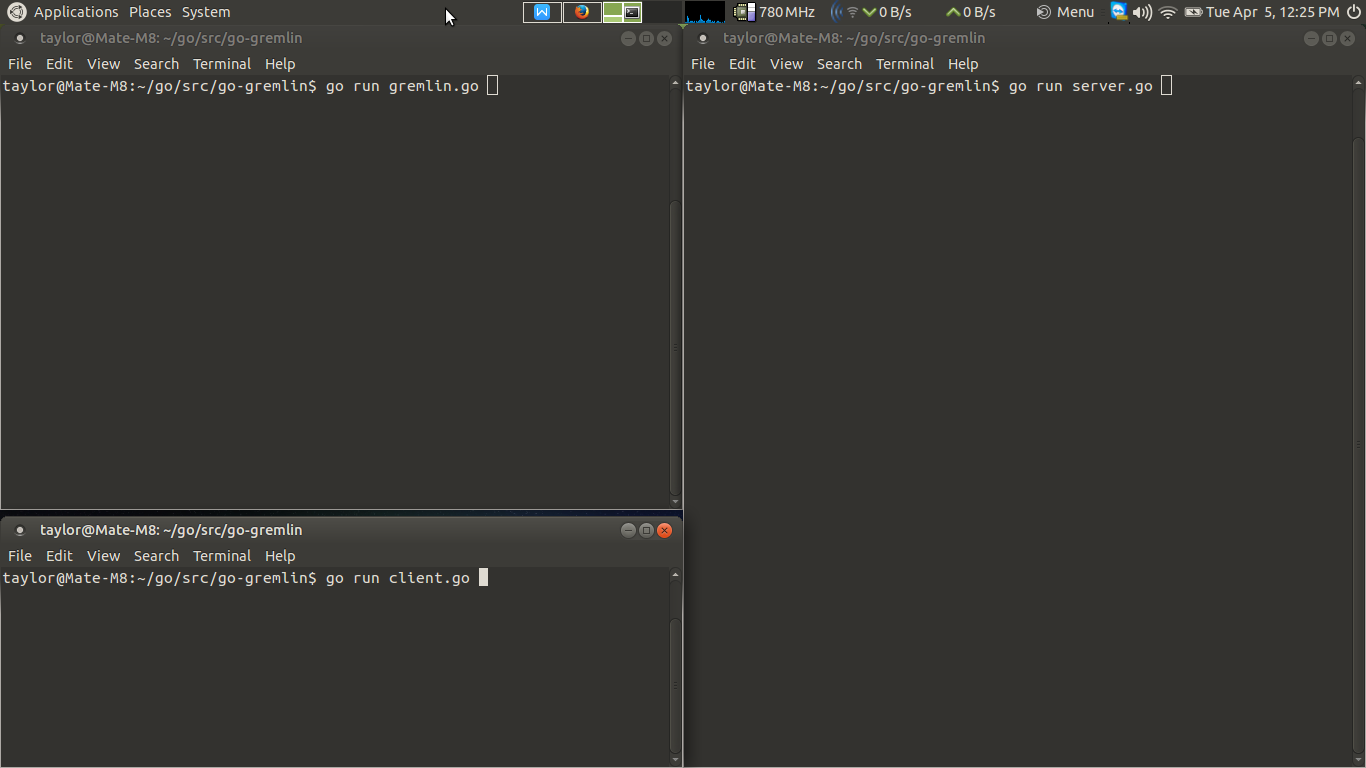
### How to Run the Program

The program requires you to have go installed. For installation instructions please refer to <https://golang.org/doc/install>.

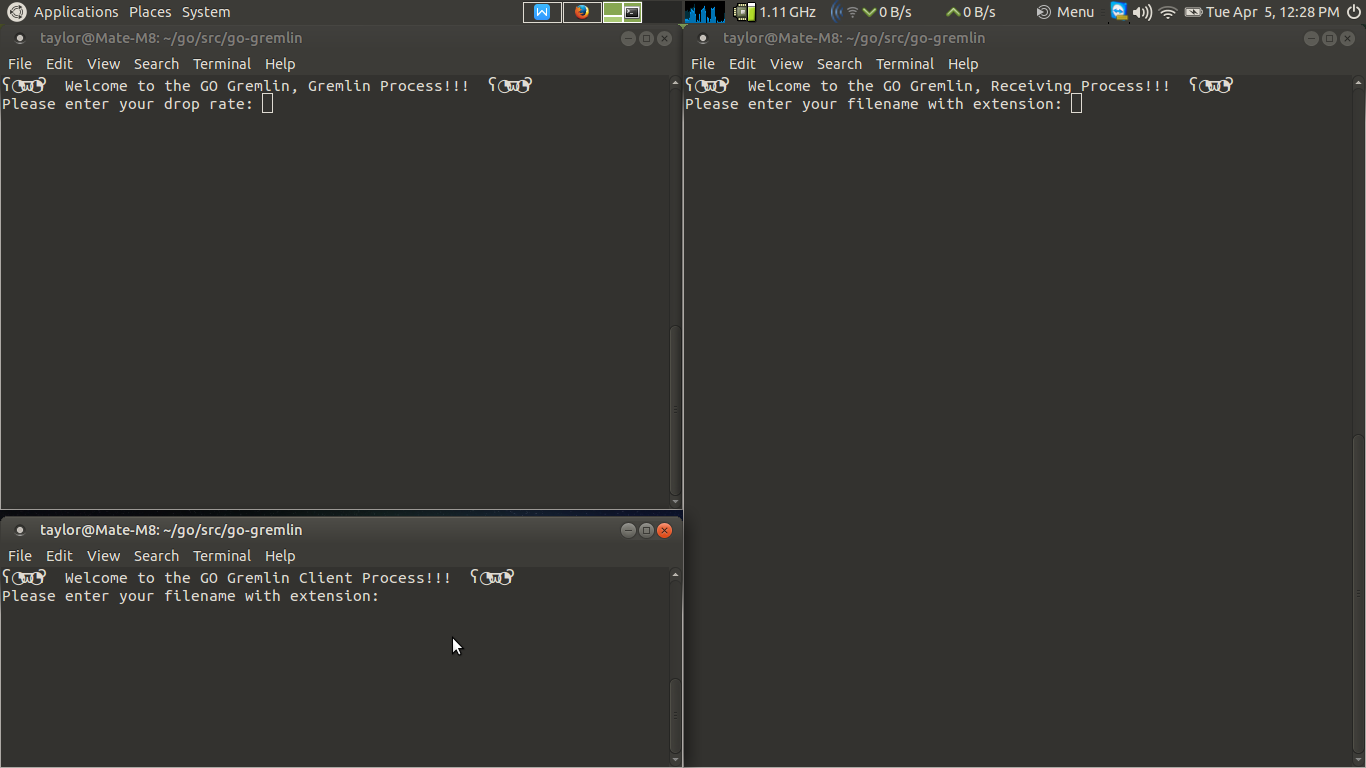
You will also need to get the files. You can get them from GitHub at the following repository, <https://github.com/chistogo/go-gremlin>, or you can get them from other means.

Once you have Go installed then you need to run the programs. You should open the programs in the following order:

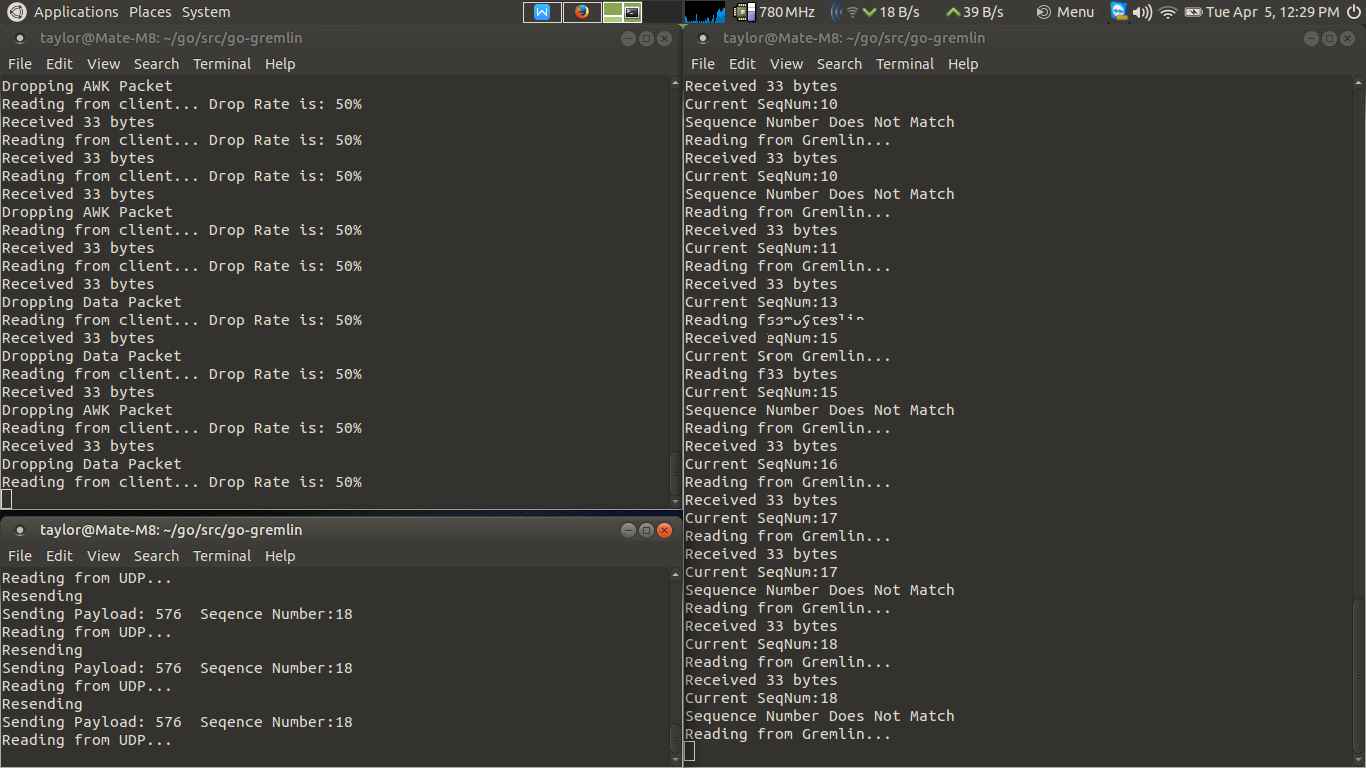
1. Gremlin.go
   1. To run this open a terminal window in the directory that contains the files and type “go run gremlin.go”
   2. After that you will be prompted for the fail rate. You will need to input a number between 1 and 100 and then press enter.
2. Server.go
   1. To run this you will need to open another terminal window in the directory that contains the files and type “go run server.go”
   2. You will be prompted for the output file name. You should type the name of what you want the new file to be including the file extension and then press enter.
3. Client.go
   1. To run this you will need to open another terminal window in the directory that contains the project files and type “go run client.go”
   2. You will then be prompted for the file name including the file extension. This file will need to be located in the same directory as the project files.



1 The Code to run the 3 parts of the program



2 Each of the programs awaiting user input



3 The Programs sending a file

### Problems Encountered

1. Packet Timeout
   1. We had problems creating a timeout for our packets. We found out that the Connection.Timeout() was deprecated and no longer worked so we had to use Connection.SetDeadline() which would create an error whenever the deadline was reached. Then we had an issue where we could not catch the error because of error.Timeout() was not found. We found a way to resolve this after asking in the GoLang IRC Channel and found out that we needed to cast the error as a net.Error.

B) If the last packet the client sent was dropped it would not complete the file

1. We made sure all of the packets would be subject to the Timeout.

### Resources Used

The GoLang Documentation Website: <https://golang.org/doc/>

The GoLang IRC Channel on Freenode: #go-nuts on irc.freenode.net

StackOverflow: <https://stackoverflow.com/>