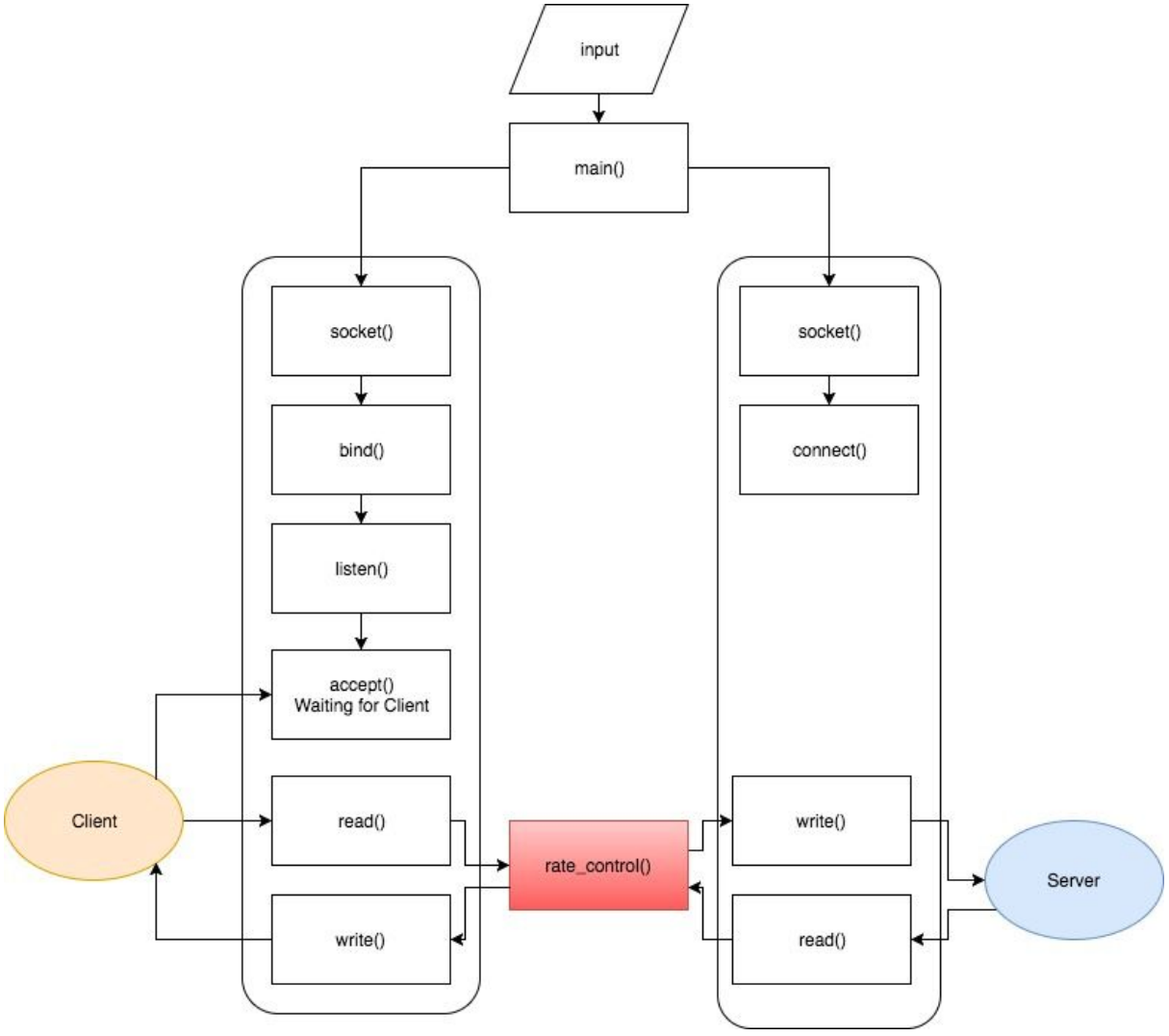
2017 ITCN Final Project Report

team_22
103070012 高于絜
103070014 潘瑋綱
103070002 蘇玫如

Flow Chart



Trace Code

- **Running of the Proxy**
  Proxy是在FTP Client與FTP Server間進行數據轉送的媒介，而這Project中所做的就是控制Server下載到Client與Client上傳到Server的傳輸速度。

- **Server and Client**
  當面對Client時，Proxy所扮演的角色是一個Server，這時create_server function中會先創一個socket，並將server地址端口進行初始化。使用bind將socket於server的地址端口進行綁定，接著對此端口進行listen，進入監聽狀態。
  而面對Server時，Proxy的角色則是一個Client，這時connect_FTP function中會先創一個socket，同樣對地址端口進行初始化，再通過connect來連接FTP Server。

- **Variables**
  Main function中會由指令讀取三個變數，分別是Proxy的IP、Port number以及所希望控制的速度。

- **Upload and Download**
  利用select descriptor 監控多個file descriptor的變化，若nready > 0則代表Client端有指令，再用FD_ISSET()巨集取得file descriptors。
  若Client中有可讀文件則代表有檔案從Client要上傳，先read from Client再write to Server。
  若Server中有可讀文件則代表有檔案從Server要上傳，先read from Server再write to Client。

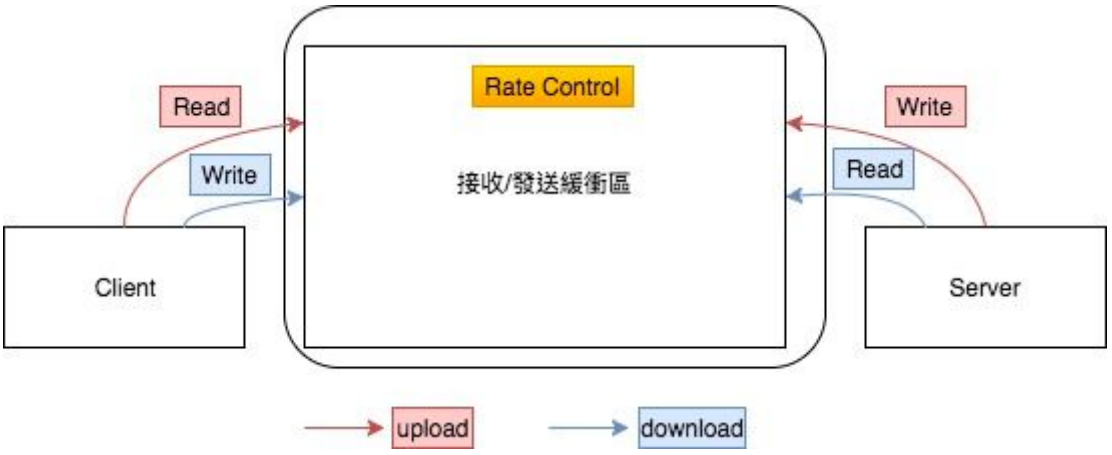
- **Transmission Rate Control**
  由於上傳時會先讀取Client端檔案再上傳到Server，因此Rate Control可以於上傳到Server前實現。下載檔案則會先讀取Server端檔案再上傳到Client，同樣也可在上傳到Client前實現Rate Control，Rate Control的方法將在下一主題中詳述。

Controlling Transmission Rate

- **File Size**
  上傳檔案時，透過Read from Client回傳的變數byte_num可得知封包大小。
  下載檔案時，透過Read from Server回傳的變數byte_num可得知封包大小。
  因此目前上傳或下載的total Byte數為每次的byte_num相加。

- **Time Calculation**
  我們使用`<sys/time.h>` 與 `<unistd.h>` 中的gettimeofday函數來取得系統時間，它的精確度可以到微秒。
  上傳檔案時，當讀取完Client的封包後，再利用gettimeofday函數來取得結束時間。
  下載檔案時，當讀取完Server的封包後，同樣再利用gettimeofday函數來取得結束時間。
  將結束時間減去起始時間則為目前已知的傳輸時間，我們所要控制的就是上傳時將讀取自Client的封包Write到Server與下載是將讀取自Server的封包Write到Client的未知時間。

- **Rate Control**



我們將封包傳輸時間、封包大小以及想控制的速度傳入rate_control function中。因為系統過快，也就是實際速度比期望傳輸速度還大時，我們就利用usleep()這個函數，讓系統delay來配合上傳及下載的速度。當上傳檔案時，藉由Read from Client可以得出已知傳輸時間以及封包大小，並根據公式"特定速度下所花費時間= 封包大小/速度"，我們可以計算出目前速度與當前速度時間的分別時間，相減之後即是想要delay的時間差。但測試後發現單單只讓系統usleep這個值是不夠的，因此我們決定再乘上一特定係數來降低平均速率，而這係數也是我們自己測試出來最符合的值。

Problems We Confronted

- 不熟悉使用方式
  一開始我們因為第一次使用proxy，因此並不熟悉應該如何讓FileZilla連上Proxy執行檔。在做了許多研究後才發現原來Terminal中執行proxy執行檔所輸入的IP及Port number就是在FileZilla中所使用的IP及Port number。

- 現實並不如想像中美好
  當我們照著Proposol的公式實作時，卻遇到了控制速率並不一定每次都如預期在範圍內。這可能是因為實際上的網路並沒有我們想像的那麼完美，速度往往會起起伏伏，根據網路的狀態而改變，造成我們以理想化公式計算出來的結果會不貼近現實，因此我們嘗試修改Rate Control的係數來使結果最佳化。

- Rate Control係數難以決定
  上面有提到usleep中只單單讓系統delay目前跟理想速度時間差是不夠的因此要乘以特定係數。而由於缺乏根據要將係數設為多少大小，我們必須不斷的try and error，從測試中記錄實驗數據並加以更改，終至決定係數數值，以達到速率控制最佳化之目的，真的耗費了不少時間與精力！

How to Run

開啟terminal並進入程式所在資料夾，編譯後執行檔案。執行指令為./proxy <Proxy IP> <Proxy Port> <Rate>

打開FileZilla並輸入相應的主機、連接埠以及使用者和密碼，開始連線後即可上傳及下載檔案。

Experimental Results

下圖顯示為 50 KB/sec [download] 的結果，經由我們不斷實驗求出的usleep()的最佳係數，而得出我們在眾多實驗中最好的結果。

Responsibility of Each Member

基本上我們都是一起約出來討論及實作，大家共同完成整個Project。

	Proposol	Coding	Testing&debug	Report
蘇玫如	★★★★	★★★★	★★★★	★★☆☆
高于絜	★★★☆	★★★☆	★★★★	★★★★
潘瑋綱	★★☆☆	★★★☆	★★★★	★★★★