Computer Network 2011

Homework 2

B97501046　資訊工程學系　三年級　李卿澄

檔案結構

b97501046_hw2.tar.gz 內包含了:

b97501046_hw2/	資料夾
– b97501046_hw2_trans.c	*Sender / Receiver source code*
– b97501046_hw2_agent.c	*Agent source code*
– b97501046_hw2.h	*Network functions and **struct** definition*
– b97501046_hw2.c	*Implementation of b97501046_hw2.h*
– Makefile	*Makefile*
– Report.pdf	*Report*（本文件）

執行程式

1. 編譯執行檔

```
$ make
```

將會編譯出 ***trans*** 與 ***agant*** 兩個可執行檔。（編譯詳細指令請見 *Makefile*）

2. 執行

1. trans

若於執行時忘記參數的設置方式與代表意義可以使用「-h」參數來取得幫助（會輸出參數說明）

1. Sender

```
$ ./trans -s -p receive_port -t target_host -tp target_port -f
filename [-o new_filename] [-a agent_host -ap agent_port]
```

參數說明:

-s	作為 Sender 執行
-p *receive_port*	設定收取 Ack 封包的 port
-t *target_host*	設定傳送檔案目的地
-tp *target_port*	設定傳送檔案目的地 port
-f *filename*	設定欲傳送的檔案名稱
-o *new_filename*	設定送至目的地後的新檔名（Optional）
-a *agent_host*	設定使用 agent 及其位置（Optional，需搭配-ap）
-ap *agent_port*	設定 agent 的 port（Optional，需搭配-a）

2. Receiver

```
$ ./trans -r -p receive_port [-o filename]
```

參數說明:

-r	作為 Receiver 執行
-p *receive_port*	設定收取檔案的 port
-o *filename*	設定將收到的檔案將使用的檔名（Optional）

2. **agent**
   若於執行時忘記參數的設置方式與代表意義可以使用「-h」參數來取得幫助（會輸出參數說明）

```
$ ./agent -p listen_port -l loss_rate
```

   參數說明：

-p *listen_port*	設定 Agent 所欲收取封包之 port
-l *loss_rate*	設定封包丟失率

3. 程式執行順序

   建議先開啟 Receiver、再視情況決定是否使用 Agent，最後才開啟 Sender，由於設計中包含了結束封包（Fin），因此執行中的程式們在工作完成後會自動關閉。

架構設計

參數設計

參數的設計與每個程式所需要的資訊息息相關，依照 trans 作為 sender 或 receiver、agent 等特性，我讓這三者都各有一個-p 參數，各自有一個用來 listen 的 port，只要有訊息（封包）要給任一支程式，只要把訊息送到那個 port 就對了。如 agent 的 port 只收欲轉送的封包，sender 只收 ack 封包，receiver 只收 data 封包。

而 agent 的功能與封包結構的設計同樣影響了參數，我認為，若作為一個 receiver，在程式啟動時只應該需要設定 listen 的 port，等待資料（無論從哪裡來）送達，因此有沒有 agent 是不應該影響 receiver 的設定，如果有透過 agent 傳輸，那麼 agent 的資訊應該寫在封包裡面，而不是一開始就告訴 receiver 有個 agent 存在。

再回來看 agent，他只負責轉送封包，只要將 from where to where 的訊息寫在封包裡面，由 agent 解讀後再轉送封包即可，只需要設定好 listen 的 port，這樣的資訊就已經夠多了。那 loss rate 是因為要模擬掉封包的狀況才需要的參數。

另外，根據需求要能在 sender 與 receiver 兩端皆可更名，因此我在兩端都加上了-o 參數，若 sender 和 receiver 皆無設定更名，則會按照原本的名稱，若其中一端設定，則就採用其設定，若兩端接設定更名，那麼則以 receiver 的設定為優先。

運行流程

Agent

在我的設計之中，agent 與其他兩者都相同，在接收封包都是使用 *select* 這個 function 來達到 non-blocking I/O 的效果。Agent 在收到一個封包之後會先看他是不是個 ack/fin 封包，若是則不阻攔，直接轉送；若是 data 封包，則先假定在某個固定範圍內的自然數出現的機率皆相等，若隨機挑選的數字比設定的封包流失率還要大，那就讓封包轉送，反之則 drop。

因為我在 bind 的時候並不是指定特定的網路介面，因此在發送封包之初，沒辦法拿到自己的 address，但是收取此封包者可以取得 address，若 agent 發現這個封包的 From 部份沒有設定（我定義沒有設定是被我設定為 0.0.0.0），那就把 agent 取得的 address 寫進去。

Sender

前面有提過使用到了 *select* 這個函式，除了 non-blocking 的好處之外，他還可以順便連 timeout 都一起設定。在進入 main loop 之後，首先是看能不能送出封包，接著就把封包都送出去，然後開始等 ack，每次收到 ack 就檢查是不是送出去的封包都得到 ack 了，若都得到則進行下一批傳送，如果等到

select timeout 的話那就知道這一批送出去的封包有的沒有得到回應，那就可以重傳了。

而由於封包有時候需要重新傳送，順序會有點錯亂，因此在讀取檔案製作封包時，是使用了 *lseek* 函式配合 seq#來直接讀檔，避開連續讀取可能會遇到的錯亂狀況。

另外，在所有 data 封包都傳送完畢並都收到 ack 後，sender 才會發送 fin 封包，並等到 ack fin 才會結束程式。

Receiver

Receiver 收到的封包會先放入緩存區中，待緩存區滿後在寫出至檔案。而每一個收到的封包，都會先檢查其 seq#是不是在目前緩存區可接受的儲存區間，唯有在儲存區間內且尚未寫入的封包會被寫至緩存區內。緩存區大小依據規定是 32 個封包，因此只要搭配清出緩存的次數就能計算目前應該接收的 seq#區間。而每當收到一個封包後，確定是新收下的或是已經收過（會出現已經收過的封包代表 sender 沒有拿到該封包的 ack）的都會回覆一個 ack 封包，回覆的目的地是從收下的封包中取得，有可能傳回給 agent，若未經 agent 也可以直接傳回給 sender。

封包結構

我的封包是以一個 unsigned char array 的形式呈現，將整個 char array 寫出去就算送出去了。並搭配上了幾個 function 以及 *enum* 幫助程式存取。封包依序存放的資訊有：

- 出發地 ip address（若設為 0.0.0.0 代表直接由 sender 發出，其 address 可由收端取得）
- 出發地 port
- 目的地 ip address
- 目的地 port
- 封包樣式（filename/data/ack/fin）
- agent port（agent address 可以由收端取得）
- seq#
- data length
- data（1024bytes）

其中 ack/fin 封包不包含 data length 和 data 區段。

程式撰寫

UDP Socket

UDP socket 的使用與 TCP socket 略有不同，一般要作為一個 server 端的作法就是 *socket*、*bind*、*listen*，然後 *accept* 一個連線（並得到一個新的 file descriptor），這是 TCP socket 的作法，而在 UDP socket 只要直接用 listen 的 fd 就可以 read/write 了。

recvfrom

在 TCP socket 的情況下，使用 accept 函式可以取得發送端的 address 資訊，可是 UDP socket 又不用 accept，那要如何取得 address 資訊呢？只要使用 recvfrom 這個函數即可取得，非常方便呢！

函式切割

程式越寫越大，自然就要把一些一直用到的東西拿出來，變成一個 function，因此在本次作業之中有另

外的 b97501046_hw2.h 與其實作 b97501046_hw2.c，包含了製作封包 header 以及從封包取得資訊的 function，而至於一些 networking 相關的程式我沒有包成 function 則是是擔心其設定多樣，若製成 function 可能設定更不方便些，所以才留下直接的設定。

令封包設計富有彈性的是除了把封包的操作盡量包在 function 裡之外，就是使用 enum 來定義一些封包常使用的常數，讓我們只需要修改 b97501046_hw2.h/.c 而不用修改到 trans 或是 agent，也不必擔心修改後會沒辦法運作，因為已經有了適度的 decoupling 了。