Computer Network 2011

Homework 3

B97501046　資訊工程學系　三年級　李卿澄

檔案結構

b97501046_hw3.tar.gz 內包含了:

b97501046_hw3/	資料夾
– b97501046_hw2_pingtalk.cpp	*Pingtalk main program*
– cnping.h	*Ping object*
– cnping.cch	*Implementation of Ping object*
– Makefile	*Makefile*
– Report.pdf	*Report*（本文件）

執行程式

1. 編譯執行檔

```
$ make
```

將會編譯出名為 ***ping*** 的可執行檔。（編譯詳細指令請見 *Makefile*）

2. 執行

1. Ping

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）

```
$ ./ping [hostname|ip] -c [request times] -s [buffer size] -t [ttl]
```

參數說明:

hostname or ip	要 ping 的目標 hostname 或 ip address
-c *request times*	發送 icmp echo 封包的次數，預設 10 次
-s *buffer size*	設定 icmp data 區塊的大小，預設 56 bytes
-t *ttl*	設定封包的 TTL，預設 20 秒

2. Pingtalk

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）

```
$ ./ping [hostname|ip] -m 1
```

參數說明:

hostname or ip	設定目標 hostname 或 ip address
-m *mode*	設定模式，mode=1 為 Pingtalk 模式

進入 Pingtalk 模式後，程式會提示指令下達方式並得到一個程式的 ***ID***:

```
B97501046 CNHW#3 PingTalk
PingTalk mode
```

```
My ID=7038
type `t {n|e} [receiver id] [text](no ws)' to send text msg
type `f {n|e} [receiver id] [file] [new name]' to send file
type `q' to quit this program
```

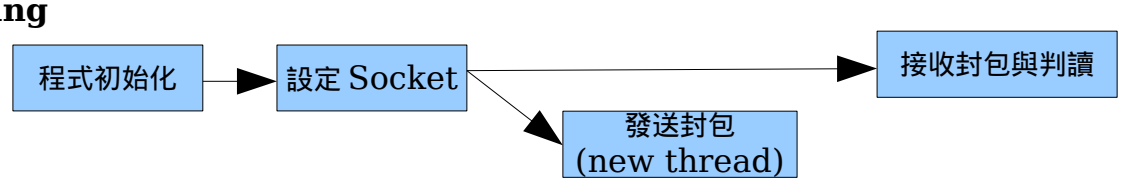
1. 文字訊息傳輸
   若欲送出未經編碼之文字訊息，只要輸入
   t n [目標之 ***ID] [***文字訊息***]***
   要特別注意的是文字訊息不可包含空白或換行等字元。而要以 base64 編碼的話只需要改成
   t e [目標之 ***ID] [***文字訊息***]***
   即可。

2. 檔案傳輸
   若欲送出未經編碼的檔案，只要輸入
   f n [目標之 ***ID] [***檔案名稱***] [***收端檔案名稱***]***
   即可，若欲將傳送的內容以 base64 編碼只要改成
   f e [目標之 ***ID] [***檔案名稱***] [***收端檔案名稱***]***

3. 結束程式
   輸入
   q
   就可以結束程式。

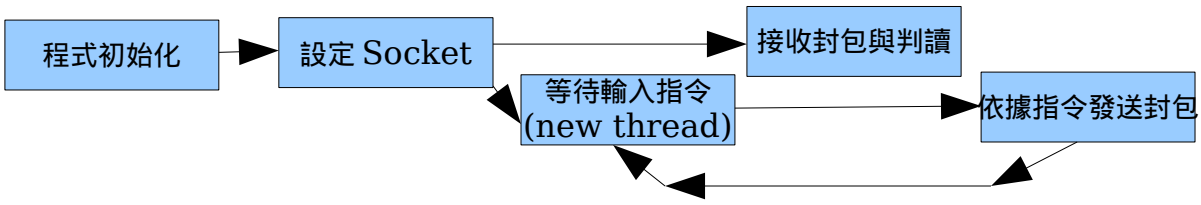
運行流程

Ping



程式根據所給定的參數進行初始化（Ping->init()），接下來設定 socket（像是設定 TTL）（Ping->setting()），然後進入 Ping->main_loop()，在 main_loop 中根據先前設定呼叫不同的 loop function，在這邊是呼叫了 ping_main_loop，進入 ping_main_loop 之後會開出一個 thread 專門負責發送 ping 封包，而 main thread 則負責接收封包並且檢查協定、Checksum 等以及解析 ICMP header 的訊息。

Pingtalk



在進入 Ping->main_loop()之前的流程與 Ping 相同，但進入 main_loop 之後依據 mode 而進入了 pingtalk_main_loop()，於此時會開出一個新 thread，此新 thread 負責等待 user 輸入指令，並依照指令發送封包，發送完畢後繼續等帶下一個指令的輸入；而 main thread 則負責收取封包（就是一個 while(1)+select，一直等，等到 user 決定讓程式結束為止）並檢查、解讀之，解讀後顯示於終端機上。

程式撰寫

Ping source code 參考

由於這次的作業與我們常常使用的一隻小程式 ping 有很大的關聯，又想說我自己平時都是在 ubuntu 下使用 ping，說不定可以找到原始碼來參考一下，果然只要 sudo apt-get source iputils-ping 就可以得到 ping 的原始碼，其中以參考 ping.c、ping_common.h 與 ping_common.c 三個檔案獲益最多。

struct icmphdr & struct iphdr

在參考 ping 原始碼的時候得到（我覺得）最重要的兩個 struct 就是 icmphdr 和 iphdr 了。在這裡有個我覺得蠻有趣也挺重要的技巧。在 icmphdr 和 iphdr 兩個 struct 之中分別定義了在 ip 和 icmp 封包的 header 中該有哪些資料、他們的 type 與次序。因此當我們利用 recvfrom 收到了 data 並放進 buffer 之後，我們可以很笨拙的以 buffer[]來存取、或判斷，這種方法可能還要面對許多的 bytes 要一起看成某個資料的狀況，那樣比較不方便。我們可以宣告一個 struct icmphdr* icmp_header，接下來令 icmp_header = (struct icmphdr*)buffer，這樣我們就可以使用 icmp_header 來方便的取得 buffer 的內容如利用 icmp_header->type 來取得 icmp packet header 的 type 資訊了。

sendto & recvfrom

這次封包的收送是開一個 Raw Socket，然後藉此直接送出 icmp header packet 來做一些事情，送出與接收分別使用了 sendto 與 recvfrom 兩個 function。在送出的部份，最小只要弄個 struct icmphdr，填好內容算好 checksum 就可以送出去了，這樣最小是送出 8 bytes；收封包的話，收到的封包會有一個 ip header 在最前面，接下來才是 icmp header，所以可以先用 struct iphdr* 找 ihl，然後乘上 4 就知道 icmp header 的開頭是在收到的資料中的位置（沒意外的話是第 20 個 byte 開始是 icmp header），接下來只要再用 struct icmphdr*就可以分析 icmp packet 的內容了。

ICMP header ID

我們可以發現，在 ping localhost 的時候會收到兩個 packet，一個是 request，另外一個是回覆，我們可以利用其 type 來區辨；那當我們有兩個 pingtalk program 開在同一台電腦上時，要進行 pingtalk 每個 process 就會收到一堆 icmp packet，這就要利用 icmp header 的 id 輔以 type 來決定什麼是當前這個 process 該處理的 packet。我自己的作法如同 ping 原始碼一樣，把 pid 作為 id，並在程式開始時提示使用者使使用者可以對不同 id 的 process 進行 pingtalk。

RTT 計算

RTT 計算有許多方法，比方說可以先取得時間，然後放進 icmp packet 的 data 區，然後收回來的時候再跟系統拿一次時間，然後跟 packet 內的時間相減得到 RTT。但是因為我們要可以調整 data 區的大小（buffer size），若將 buffer size 設定為 0 那不就沒有地方放時間資訊了？所以我是在程式內留了一段記憶體給送出的封包記下送出時間，並以 seq number 作為 index，等收到回覆就能再跟系統要一次時間，然後根據 seq number 找出送出時間，然後算出 RTT。

pthread in C++ class

另外一個值得一提的是，由於提到了 ping 每秒送一個封包出去，因此我想到的實做方式是開一個新的 thread，將 send 和 receive 分開，因為我是使用 c++來寫這次作業的，我把 send 和 receive 都寫成是 Ping 這個 object 的 private function，那要用 pthread_create 的話就會遇到 function 型別不符的問題，而解決之道是寫一個代理 function，讓他成為 Ping object 的 friend，然後將此 function 作為 thread 的 function，在 pthread_create 之時將物件自身(this)作為參數傳給該 function，在 function 之中再呼叫 send function 來達成把 send 作為一個 thread 的目的。

Pingtalk data 格式

為了讓程式可以利用 icmp packet 傳遞訊息與檔案，我們要把訊息或檔案置於 icmp data 區域。我們必須添加一些資訊以利於檔案及訊息的傳送。下面是我添加在 data 區最前面的內容（struct icmpdata）：

Type (1byte)	Encode (1byte)	From (2bytes)
Len (2bytes)		Flags (2bytes)
Offset (4bytes)		

Type 用來區分是傳送檔案還是文字訊息，Encode 紀錄編碼種類（目前只有無編碼或 base64），From 紀錄發送者的 id，讓收端可以知道訊息來自哪裡，Len 紀錄 struct icmpdata 之後有多少 bytes 真正的 data，Flags 目前訂有「這是一個發送檔案名稱的封包」、「這是一個內容為檔案內容的封包」和「本次傳送就到這個封包為止的封包」三種，Offset 是該資料在檔案內的 offset 是多少的紀錄。

Template Method

這次的程式撰寫稍稍利用了 Design Pattern 中 Template Method 的概念，但沒有完整的利用物件導向的特性來達到效果。這次作業可以分為 ping 和 pingtalk 兩大部份，而我們可以在既定的程式執行流程下有不同的實做方式（固定「初始化 → 設定 → 分為收與送兩個 thread」的流程，收、送可有不同實做方式）。程式都是以 Ping->init，Ping->setting，Ping->main_loop 的固定流程運作，在 main_loop 內依據不同的 mode 選擇不同的 main_loop 實做。若要作到更細緻應該是將 Ping 作為一個 Virtual Class，定義 main_loop 以及收、送功能，再來讓一個普通 Ping 和 Pingtalk class 繼承 Ping，實做其定義之 method，讓 runtime 來決定要使用哪一個 subclass。