Computer Networks

Project #1 Report

Linux Router and Socket Programming

Content

By 熊欣

2016.11

1.Set up a 2-port Linux PC Router

因爲沒有實驗室，所以主要實驗過程是藉助助教在實驗室搭好的 Router 完成實驗，相關軟件安裝實在自己筆電虛擬機上的 Linux 里完成的。

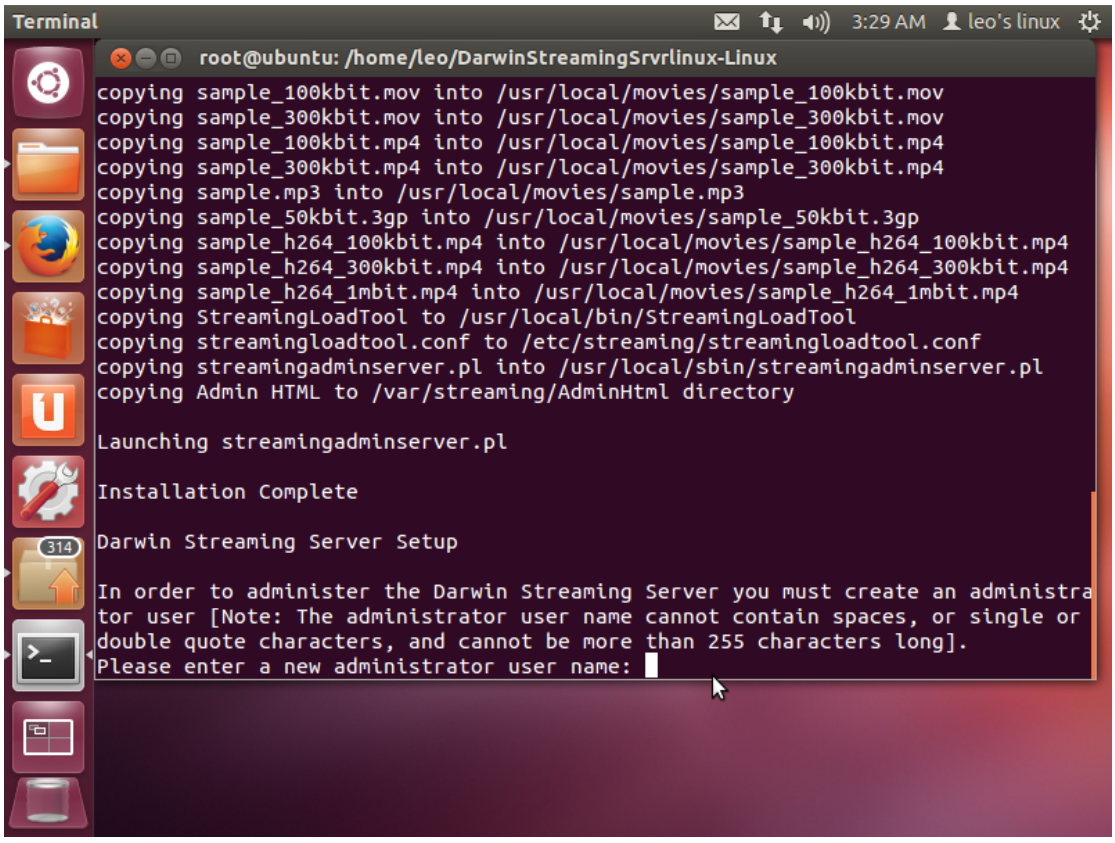
2.Install Video Streaming Server Darwin

2.1.解壓縮並進入壓縮後檔目錄

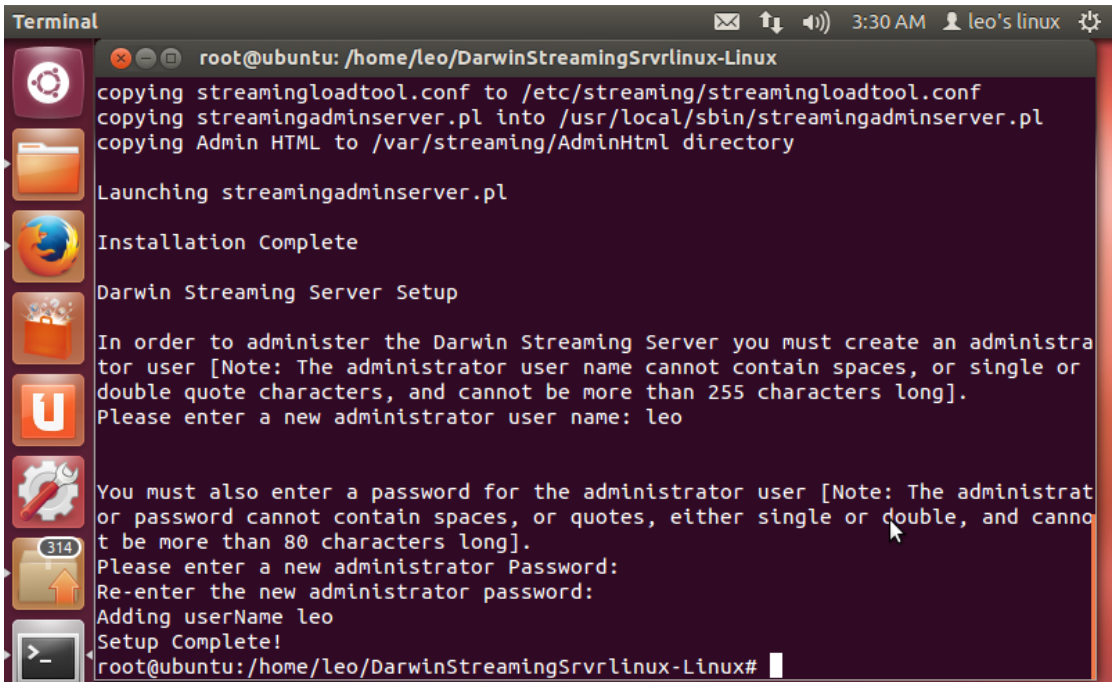
tar zxf DarwinStreamingSrvr5.5.5-Linux.tar.gz

cd DarwinStreamingSrvrlinux-Linux

./Install



2.2 建立新的用戶名和密碼

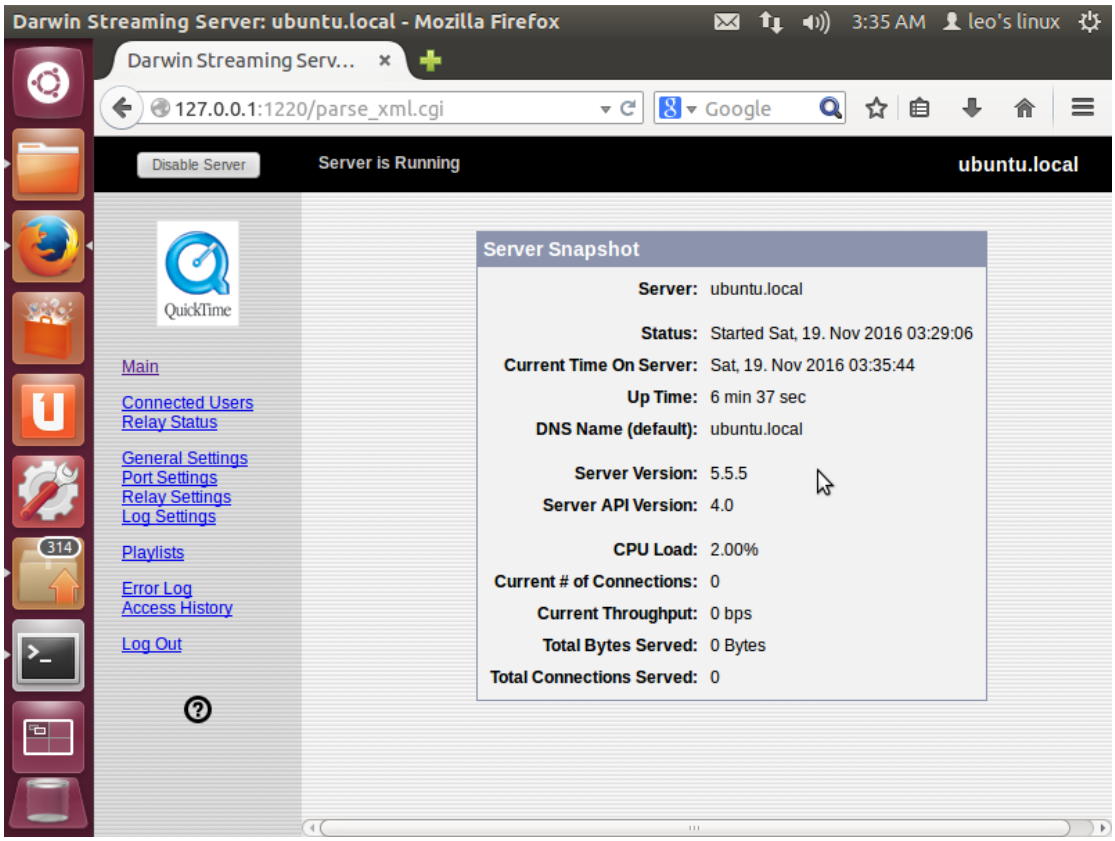


2.3.依次輸入以下命令啟動 Darwin

sudo /usr/local/sbin/DarwinStreamingServer

sudo /usr/local/sbin/streamingadminserver.pl

流覽器打開 http://127.0.0.1:1220/



3.Install VoIP Asterisk

3.1.解壓縮並進入壓縮後檔目錄

Tar zxf asterisk-1.8-current.tar.gz

cd.asterisk-1.8.32.3

3.2.依次執行如下命令，安裝必要檔

i. sudo apt-get install linux-headers-$(uname -r) build-essential automake autoconf bison flex libtool libncurses5-dev libssl-dev subversion svn-buildpackage

ii. sudo apt-get install cvs libgsm1 libgsm1-dev

iii. sudo apt-get install sysvconfig

iv. sudo apt-get install libxml2 libxml2-dev

3.3.make clean

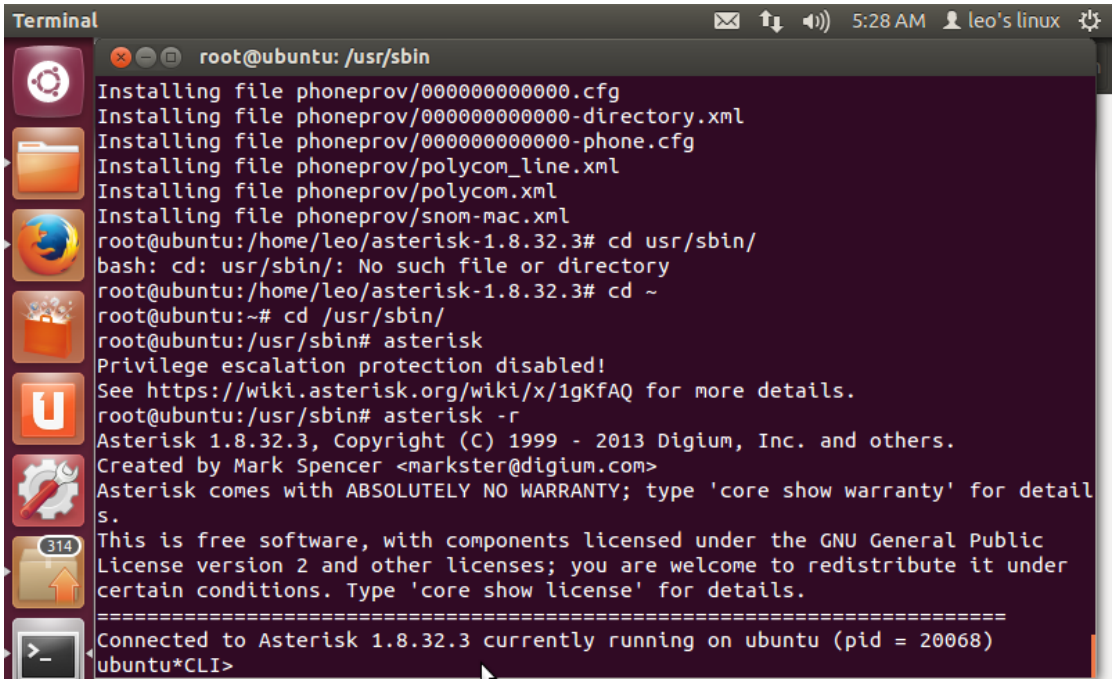
3.4. ./configure

3.5.make

3.6.make install

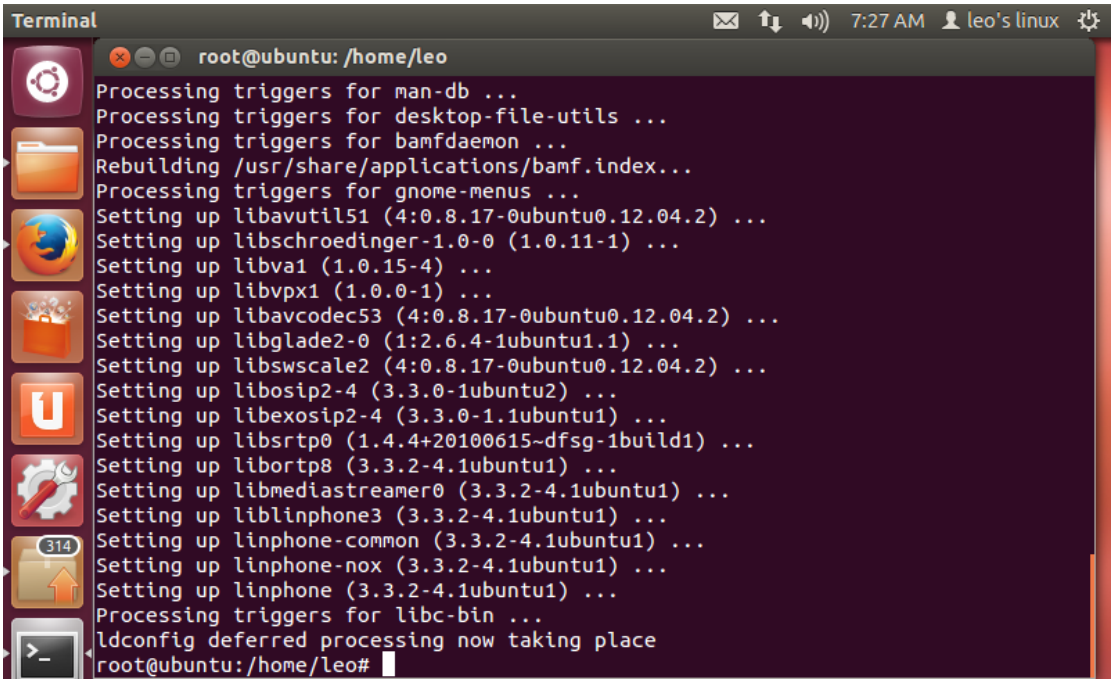
3.7 make samples

3.8.進入/usr/sbin/目錄下，輸入 asterisk 命令，再輸入 asterisk-r 完成連接

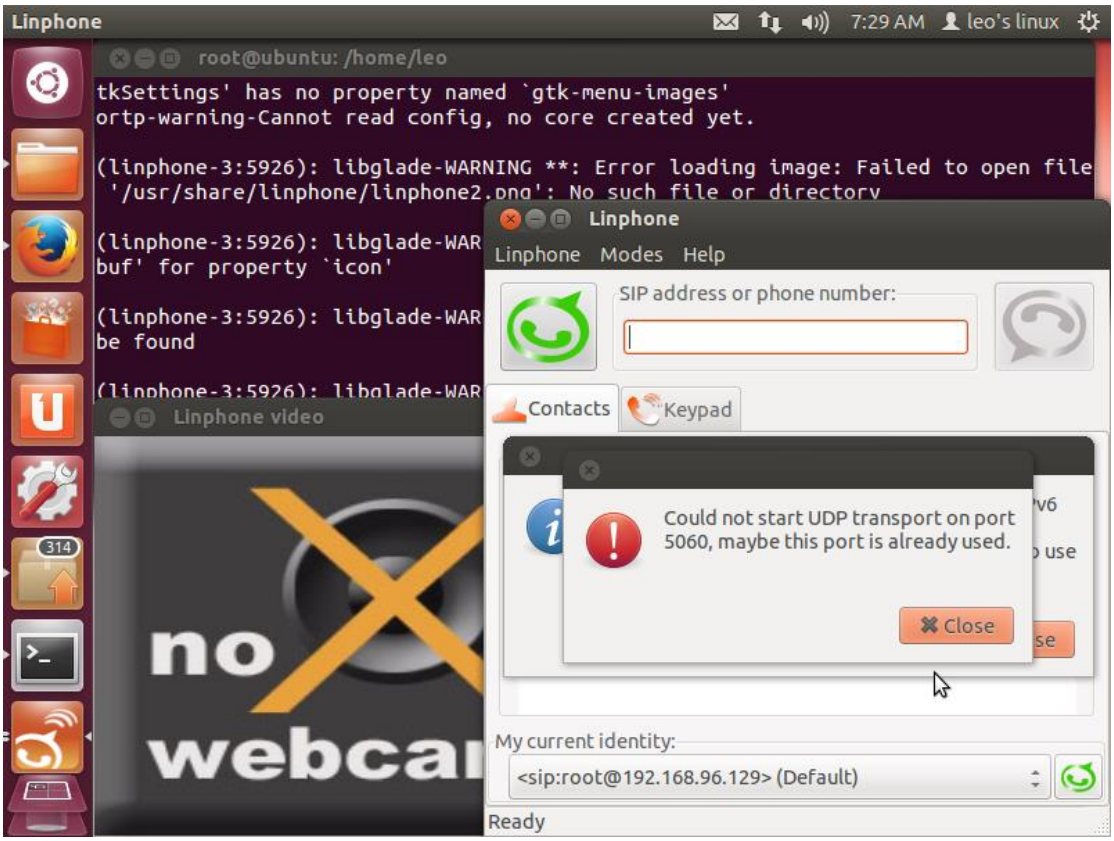


4.Install Linphone

4.1.sudo apt-get install linphone



4.2.linphone-3(因為在自己筆電上安裝還沒有連接到 router，所以還不能打電話。

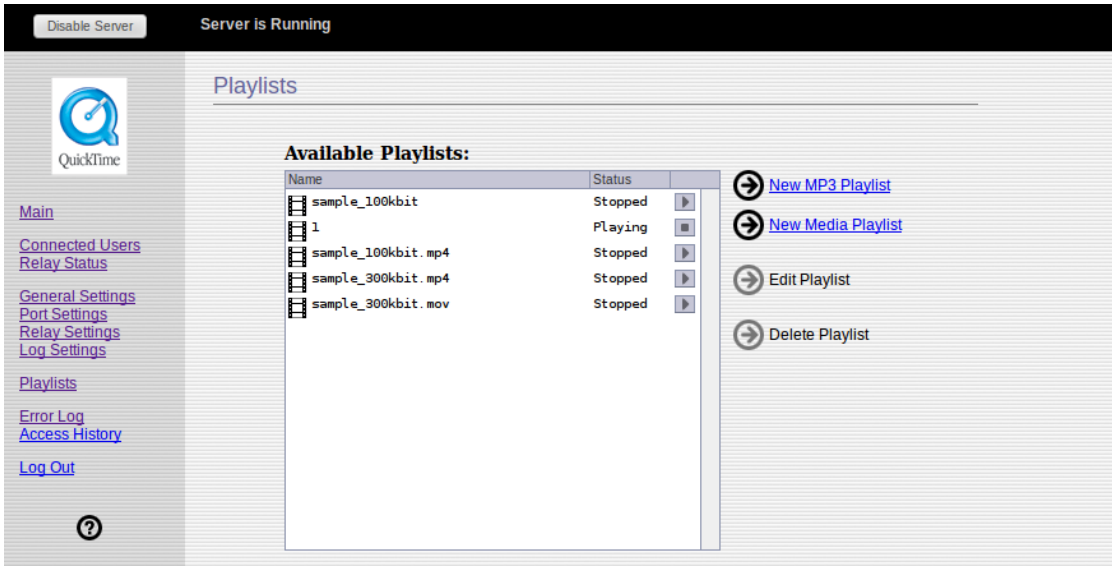


5.Test Darwin

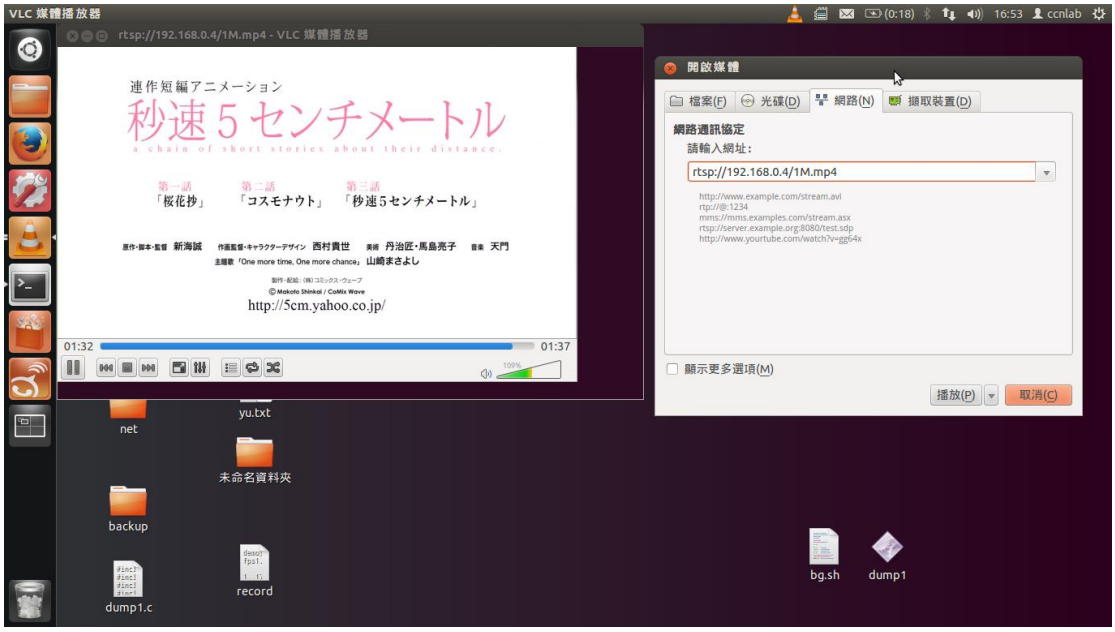
5.1.client 端與 server 端互 PING 成功，server 端与 router 互 PING 成功，client 端与 router 互 PING 成功

```
silvia@silvia-VirtualBox:~$ ping 192.168.0.2
PING 192.168.0.2 (192.168.0.2) 56(84) bytes of data.
64 bytes from 192.168.0.2: icmp_req=1 ttl=64 time=0.427 ms
64 bytes from 192.168.0.2: icmp_req=2 ttl=64 time=0.819 ms
64 bytes from 192.168.0.2: icmp_req=3 ttl=64 time=0.748 ms
64 bytes from 192.168.0.2: icmp_req=4 ttl=64 time=0.833 ms
64 bytes from 192.168.0.2: icmp_req=5 ttl=64 time=0.826 ms
64 bytes from 192.168.0.2: icmp_req=6 ttl=64 time=0.846 ms
c64 bytes from 192.168.0.2: icmp_req=7 ttl=64 time=0.818 ms
64 bytes from 192.168.0.2: icmp_req=8 ttl=64 time=0.808 ms
^C
--- 192.168.0.2 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7006ms
rtt min/avg/max/mdev = 0.427/0.765/0.846/0.134 ms
```

5.2.server 端選擇影片播放

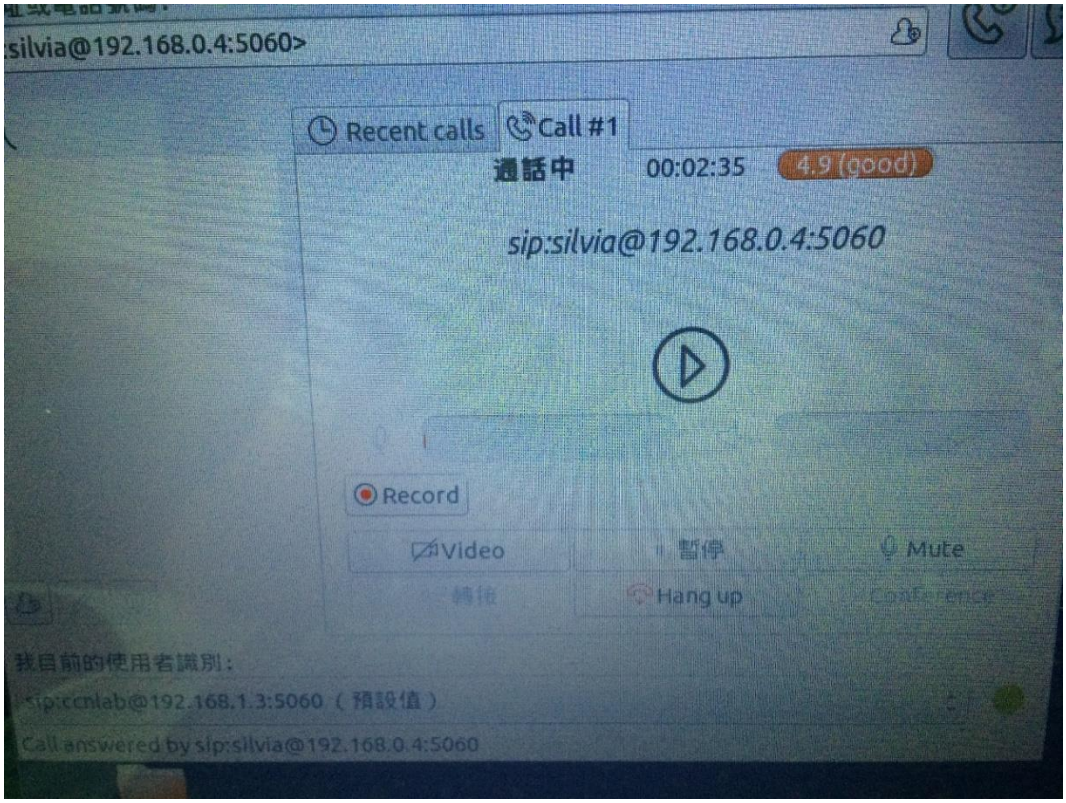


5.3.client 端 VLC 開啓媒體和網路 rtps://192.108.0.4/1M.mp4，成功播放 server 端傳來的影片

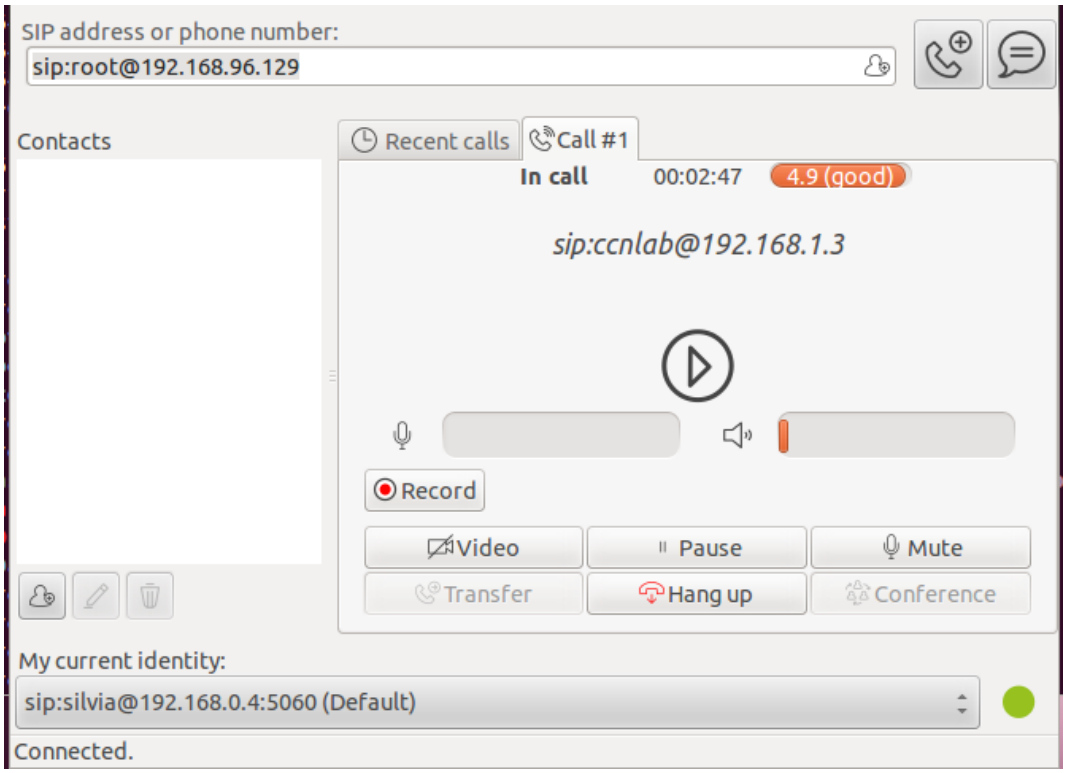


6.Test Asterisk

6.1.server 端輸入 client 端的撥號
6.2.client 端通話畫面

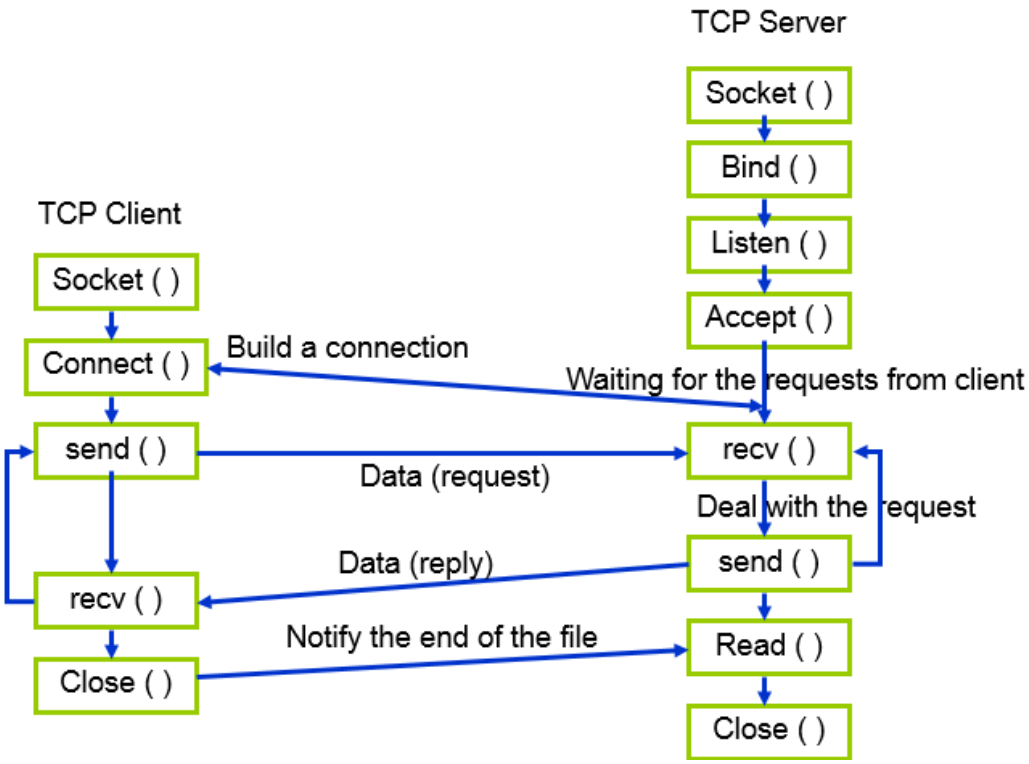


6.3.server 端通話畫面



7.Send And Receive English/Chinese Text Messages to Each Other.

7.1.write down server and client program by using TCP socket



client.c

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<sys/stat.h>
#include<fcntl.h>
#include<unistd.h>

int main(void)
{
  int sockfd,length;
  char data[1024];
  char serverIP[15];
  char buf[BUFSIZ];
  struct sockaddr_in myaddr;
  struct sockaddr_in servaddr;

  if((sockfd = socket(AF_INET,SOCK_STREAM.0))<0){
```

```
        printf("socket error\n");
        return 0;
    }
    memset((char*)&myaddr,0,sizeof(myaddr));
    myaddr.sin_family = AF_INET;
    myaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    myaddr.sin_port = htons(0);
    if (bind(sockfd,(struct sockaddr*)&myaddr,sizeof(myaddr))<0){
        printf("bind error\n");
        return 0;
    }
    memset((char*)&servaddr,0,sizeof(servaddr));
    printf("Enter Server IP:");
    scanf("%s",serverIP);
    servaddr.sin_family = AF_INET;
    servaddr.sin_port = htons(1234);
    servaddr.sin_addr.s_addr = inet_addr(serverIP);

    if (connect(sockfd,(struct sockaddr*)&servaddr,sizeof(servaddr))<0){
        printf("connect failed !");
        return 0;
    }
    printf("Send Data : ");
    scanf("%s",data);
    if (write(sockfd,data,strlen(data))<0){
        printf("write error!\n");
        return 0;
    }
    printf("Receive Data : %s\n",buf );
    close(sockfd);
    return 0;
}
```

server.c

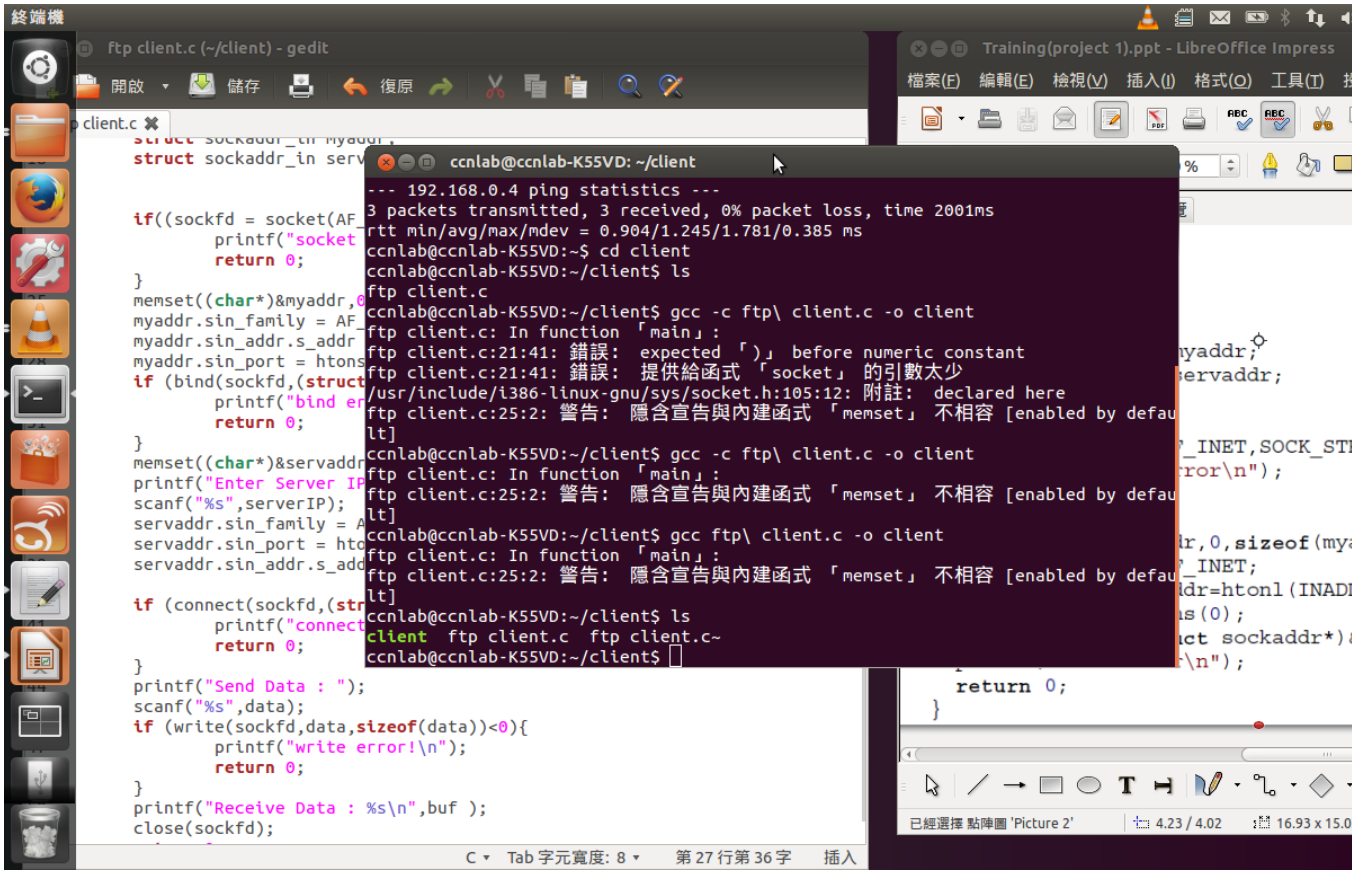
```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/errno.h>
#include <netinet/in.h> // sockaddr_in 結構所需標頭檔
#include <sys/stat.h>
#include <fcntl.h>
#include <strings.h>
#include <string.h>
#include <stdlib.h>
```

```
int main()
{
	int sockfd;
	int recfd;
	int length;
	char buf[BUFSIZE];
	struct sockaddr_in myaddr; //宣告一個 sockaddr_in 結構(註)
	struct sockaddr_in client_addr;

	/* create socket */
	if ((socket=socket(AF_INET,SOCK_STREAM,0))<0)
	{
	 perror("socket error");
	 return 0;
	}
	memset((char*)&myaddr,0,sizeof(myaddr));

	sockfd = socket(PF_INET, SOCK_STREAM, 0);
	//參數 1:IPV6 參數 2:使用 TCP 參數 3:通訊協定，一般為 0，表自動
選擇
	/* initialize structure myaddr */
	// bzero(&myaddr, sizeof(myaddr));
	//將沒有用到的結構內容清空,才不會有執行錯誤的問題
	myaddr.sin_family = AF_INET; //PF 為 IPV6；AF 為 IPV4
	/* this line is different from client */
	myaddr.sin_addr.s_addr=htonl(INADDR_ANY);
	myaddr.sin_port = htons(1234); //htons()將 2048 轉換成系統看得懂的
東西
	/* Assign a port number to socket */
	if (bind(sockfd, (struct sockaddr*)&myaddr, sizeof(myaddr)<0))
	{
	 perror("bind failed");
	 return 0;
	}
	/* make it listen to socket with max 1 connections */
	if (listen(sockfd,1)<0)//第二個參數為 server 要限制的 client 端數量
	{
	 perror("listen error");
	 return 0;
	}
	length=sizeof(client_addr);
	printf("Server start !\n");

	/* infinity loop -- accepting connection from client forever */
```

```
        while(1) //無限迴圈,1 為 true
        {
         if (recfd=accept(sockfd,(struct sockaddr*)&client_addr,&length))
         {
              perror("accept error");
              return 0;
         }
         if (length=read(recfd,buf,BUFSIZE)<0)
         {
              perror("server read error");
              return 0;
         }
         printf("Creat socket #%d
    from %s : %d\n",recfd,inet_ntoa(client_addr.sin_addr),htons(client_addr.sin_
    port) );
         printf("receive: %s\n", &buf);
         if ((write(recfd,buf,sizeof(buf)))<0)
         {
              perror("server write error");
              return 0;
         }
         memset(buf,0,sizeof(buf));
         close(recfd);
         return 0;
        }
    }
```

7.2.server 端和 client 端分別 compile server.c and client.c

7.3.client 端調試成功



7.4.server 端調試成功

```
silvia@silvia-VirtualBox:~/Desktop$ gcc ftpserver.c -o server
ftpserver.c: In function ‘main’:
ftpserver.c:21:2: warning: incompatible implicit declaration of built-in function ‘memset’ [enabled by default]
ftpserver.c:26:35: error: ‘$myaddr’ undeclared (first use in this function)
ftpserver.c:26:35: note: each undeclared identifier is reported only once for each function it appears in
ftpserver.c:48:3: warning: format ‘%s’ expects argument of type ‘char *’, but argument 3 has type ‘int’ [-Wformat]
ftpserver.c:49:3: warning: format ‘%s’ expects argument of type ‘char *’, but argument 2 has type ‘char (*)[8192]’ [-Wformat]
ftpserver.c:59:2: error: ‘rerurn’ undeclared (first use in this function)
ftpserver.c:59:9: error: expected ‘;’ before numeric constant
silvia@silvia-VirtualBox:~/Desktop$
silvia@silvia-VirtualBox:~/Desktop$ gcc ftpserver.c -o server
ftpserver.c: In function ‘main’:
ftpserver.c:21:2: warning: incompatible implicit declaration of built-in function ‘memset’ [enabled by default]
ftpserver.c:48:3: warning: format ‘%s’ expects argument of type ‘char *’, but argument 3 has type ‘int’ [-Wformat]
ftpserver.c:49:3: warning: format ‘%s’ expects argument of type ‘char *’, but argument 2 has type ‘char (*)[8192]’ [-Wformat]
ftpserver.c:59:2: error: ‘rerurn’ undeclared (first use in this function)
ftpserver.c:59:2: note: each undeclared identifier is reported only once for each function it appears in
ftpserver.c:59:9: error: expected ‘;’ before numeric constant
silvia@silvia-VirtualBox:~/Desktop$ gcc ftpserver.c -o server
ftpserver.c: In function ‘main’:
ftpserver.c:21:2: warning: incompatible implicit declaration of built-in function ‘memset’ [enabled by default]
ftpserver.c:48:3: warning: format ‘%s’ expects argument of type ‘char *’, but argument 3 has type ‘int’ [-Wformat]
ftpserver.c:49:3: warning: format ‘%s’ expects argument of type ‘char *’, but argument 2 has type ‘char (*)[8192]’ [-Wformat]
silvia@silvia-VirtualBox:~/Desktop$ ./server
Server start !
```

7.5.client 端 send text message “hello!”

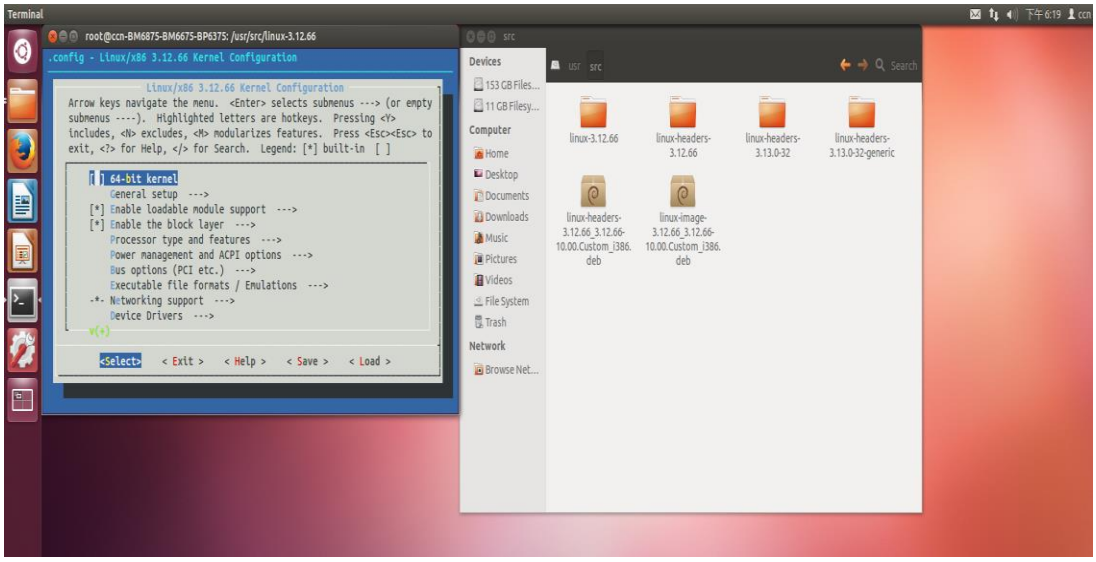
```
ccnlab@ccnlab-K55VD:~/client$ ls
client  ftp client.c  ftp client.c~
ccnlab@ccnlab-K55VD:~/client$ ./client
Enter Server IP:192.168.0.4
Send Data : hello!
Receive Data :
ccnlab@ccnlab-K55VD:~/client$
```

7.6.server 端 receive text message “hello!”

```
Creat socket #4 from 192.168.1.3 : 59111
receive: hello!
```

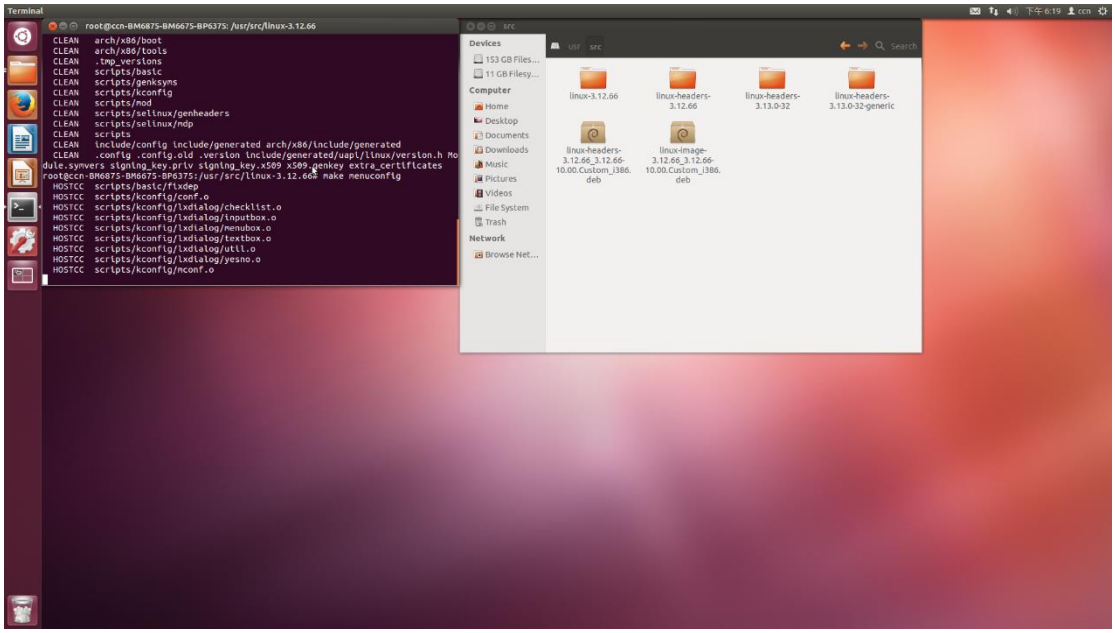
8.Install Kernel

8.1.reboot 看版本 kernel3.13.0-32-generic，去網絡上找到最接近的 Linux 版本下載

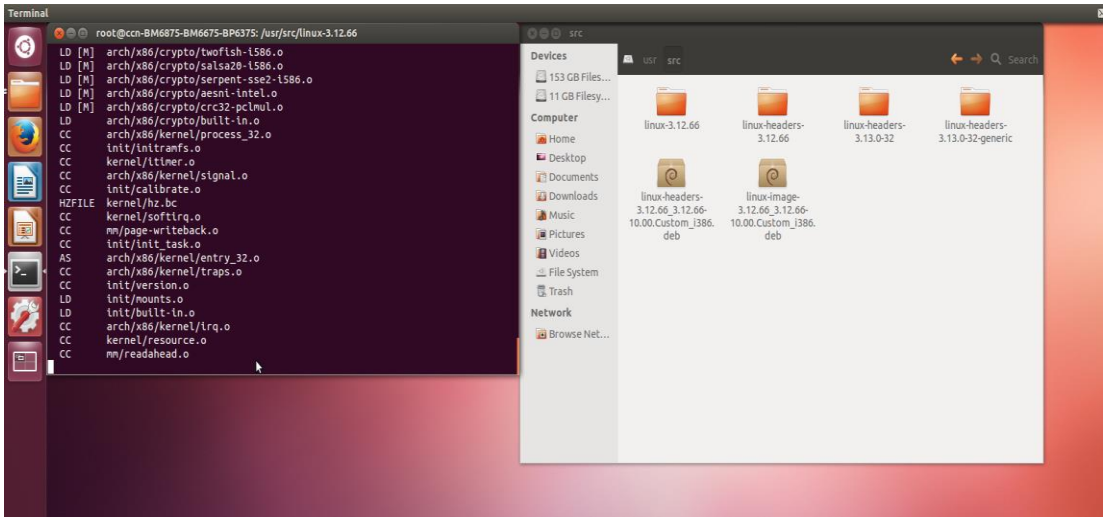


8.2.解壓縮移到 file system

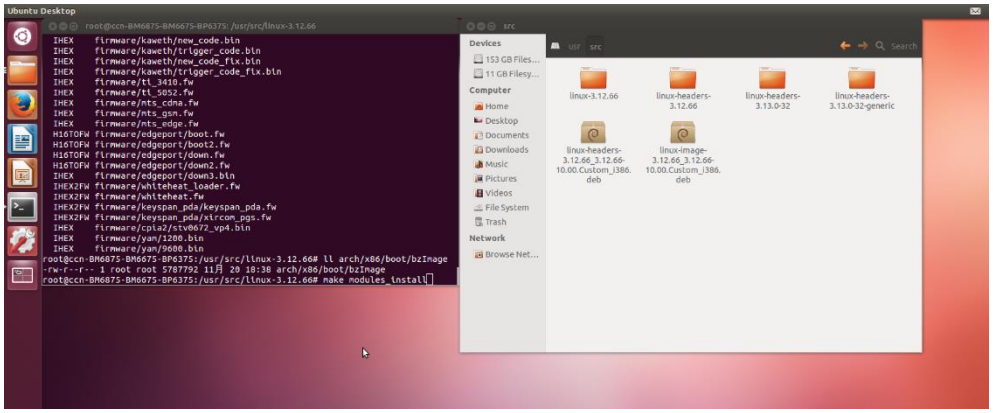
8.3.make menuconfig

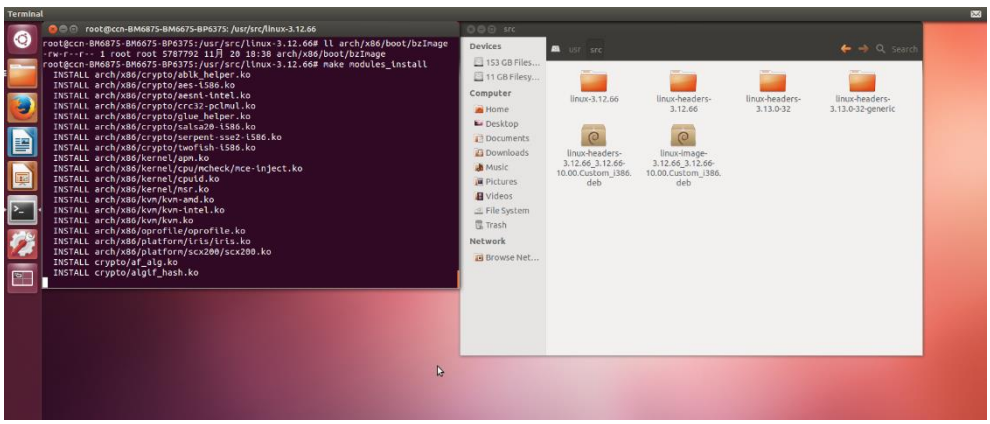


8.4.make all

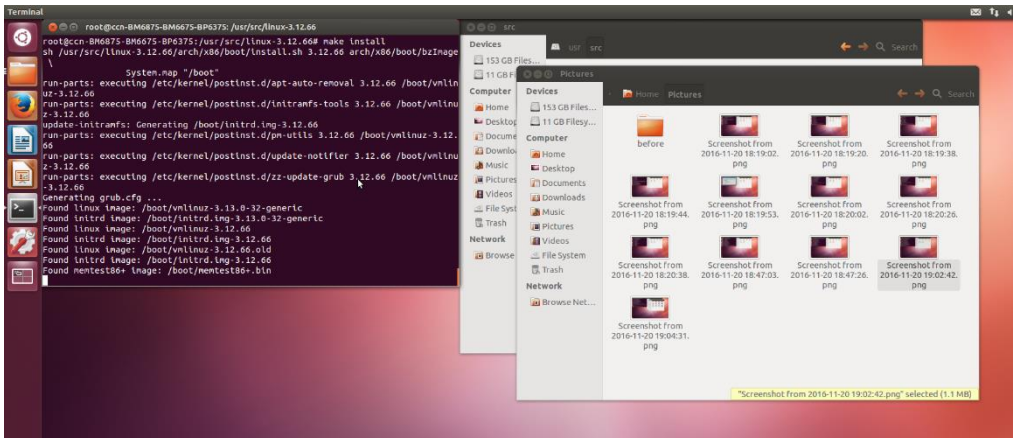


8.5.Make module install

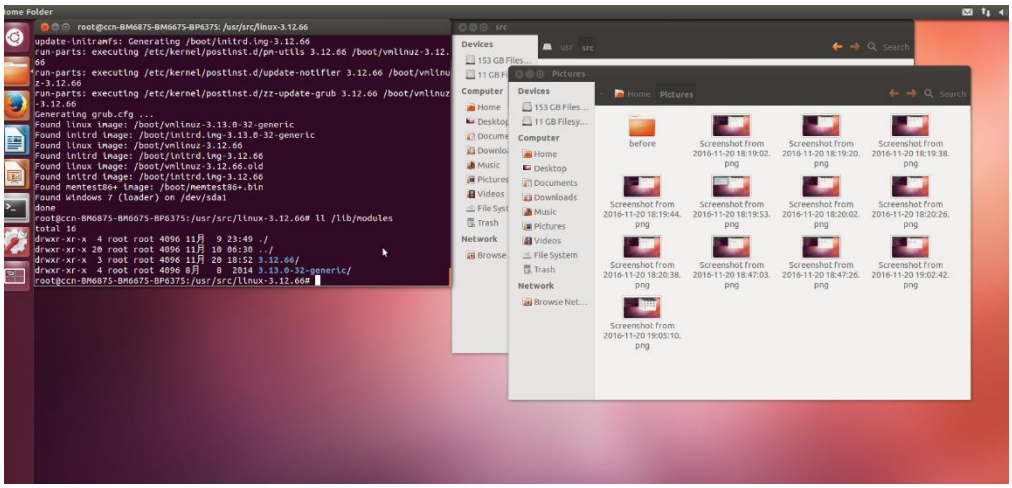




8.6.Make install



8.6.ll lib/module



8.7.安裝成功後，再 reboot 選擇 previous Linux Version 進入

9. Measure Packet Delay on Linux Router

9.1.write down backdoor program

Input_queue.c

```
int count(struct sk_buff* skb)
{
    struct iphdr *iph; //IP header
    struct tcphdr *tcph; //TCP header
    struct udphdr *udph; //UDP header
    struct timeval tv; //time struct(second & micro second)
    static int totalpacket=0;
    do_gettimeofday(&tv); //get now time
    _kernel_seconds_t intime=(tv.tv_sec*1000000)+tv.tv_usec; // intime =
    micro second
    iph = ip_hdr(skb); //get IP header

    if(iph->protocal==6){ //TCP packet
        tcph=(struct tcphdr *)((char *)iph+(iph->ihl*4));
```

```
            if(ntohs(tcph->source) == 8888) // TCP header, port number =
        8888(change to your port number)
            {
                totalpacket++;
                printk("Socket_i\t%ld\t%d\t%d\n",intime,ntohs(iph->
        tot_len),totalpacket); // time, total length, total packet
            }
        }
        return 0;
    }
```

Output.queue.c

```
    int count(struct sk_buff* skb)
    {
     struct iphdr *iph; //IP header
     struct tcphdr *tcph; //TCP header
     //struct udphdr *udph; //UDP header
     struct timeval tv; //time struct(second & micro second)
     static int totalpacket=0;
     do_gettimeofday(&tv); //get now time
     _kernel_seconds_t  outtime=(tv.tv_sec*1000000)+tv.tv_usec; // intime =
    micro second
     iph = ip_hdr(skb); //get IP header

     if(iph->protocal==6){ //TCP packet
          tcph=(struct tcphdr *)((char *)iph+(iph->ihl*4));
          if(ntohs(tcph->source) == 8888) // TCP header, port number =
    8888(change to your port number)
          {
               totalpacket++;
               printk("Socket_i\t%ld\t%d\t%d\n",outtime,ntohs(iph->
    tot_len),totalpacket); // time, total length, total packet
          }
     }
     return 0;
    }
```

Delay_time.awk

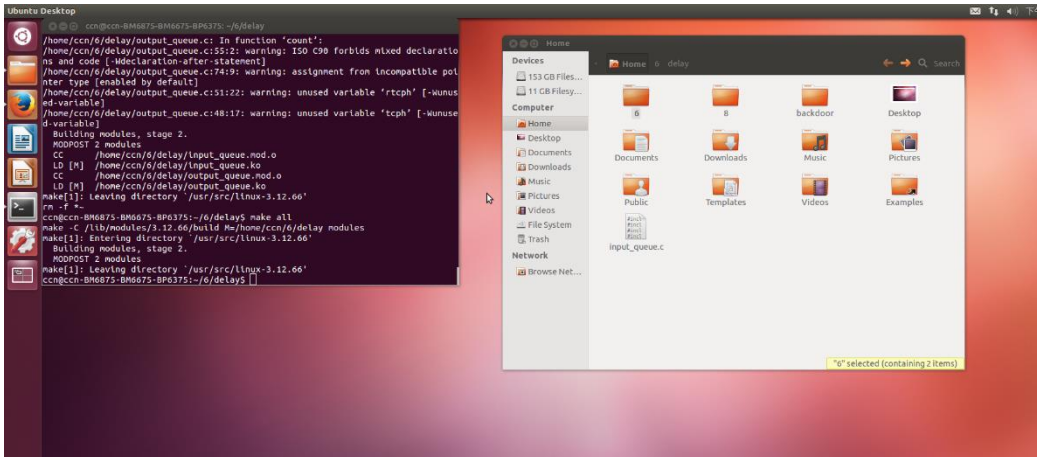
```
BEGIN{
     j=1;
     total = 0;
     total_time = 0;
     delay_time = 0;
     avg_delay_time = 0;
}
{
```

```
        IO = $2;
        time = $3;
        seq = $4;
        if( IO == "Input")
        {
            in_seq[j]=seq;
            input[seq]=time;
            j++;
        }
        if( IO == "Output")
        {
            output[seq]=time;
        }
    }
    END{
        for(i=in_seq[1];i<in_seq[j-1];i++)
        {
            delay_time=output[i]-input[i];
            total_time=total_time + delay_time;
        }
        avg_delay_time = total_time / j;
        printf("Average delay_time = %f us\n",delay_time);
    }
```

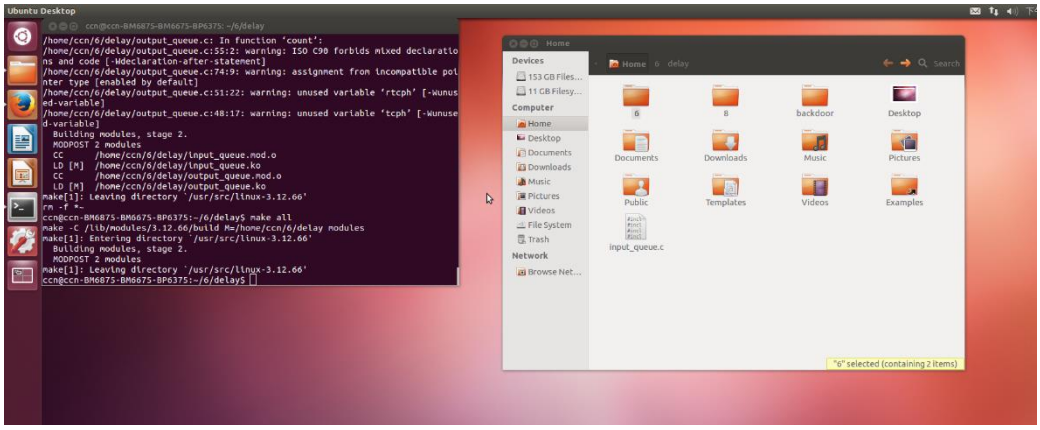
9.2.把寫好的 backdoor program 存入新建的資料夾“delay_time”，終端機進入該目錄下面

9.3.make ,把 input_queue, output_queue 的 C 檔 make 成 ko 檔

9.4.make clean



9.5.make all

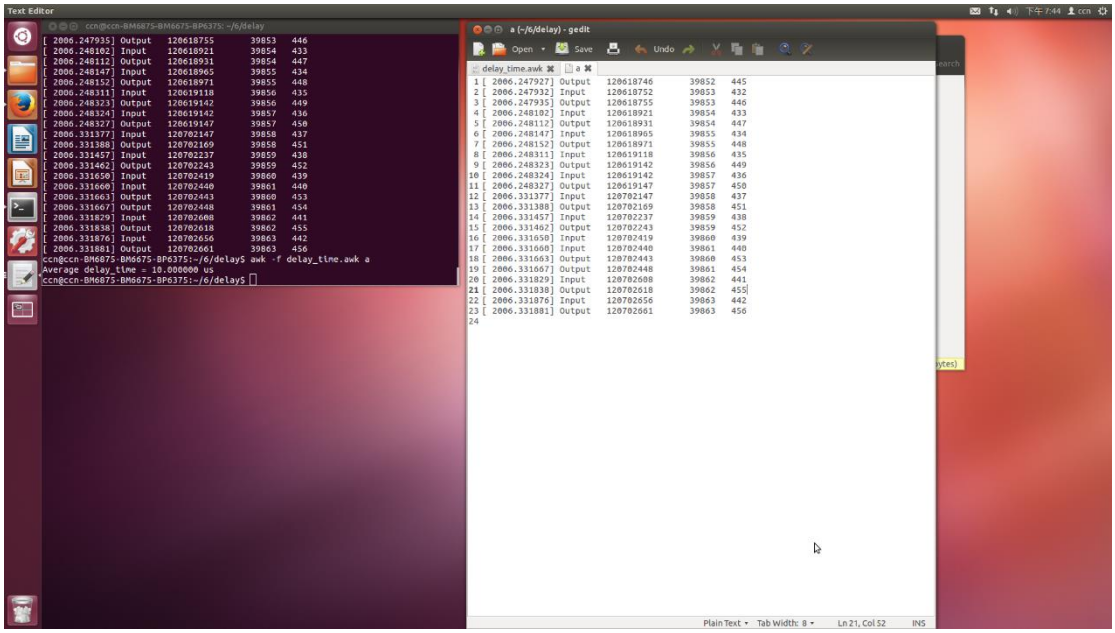


9.6 sudo rmmod input_queue，移除 input_queue.c
9.7. sudo rmmod output_queue，移除 output_queue.c
9.8.sudo insmod input_queue.ko，插入 input_queue.ko
9.9. sudo insmod output_queue.ko，插入 output_queue.ko

9.10.client 端和 server 使用 Darwin 互傳影片

9.11. dmesg,將獲取的詳細資料複製粘貼到 a 檔

9.12. awk –f Delay_time.awk a,計算 delay time



Average delay time=10.000000us

10.Measure Throughput for Darwin and Asterisk

10.1.write down backdoor program

Input.queue.c

int count(struct sk_buff* skb)

```
{
  struct iphdr *iph; //IP header
  struct tcphdr *tcph; //TCP header
  struct udphdr *udph; //UDP header
  struct timeval tv; //time struct(second & micro second)
  static int totalpacket=0;
  do_gettimeofday(&tv); //get now time
  _kernel_seconds_t intime=(tv.tv_sec*1000000)+tv.tv_usec; // intime = micro
second
  iph = ip_hdr(skb); //get IP header

  if(iph->protocal==6){ //TCP packet
        tcph=(struct tcphdr *)((char *)iph+(iph->ihl*4));
        if(ntohs(tcph->source) == 8888) // TCP header, port number =
8888(change to your port number)
        {
              totalpacket++;
              printk("Socket_i\t%ld\t%d\t%d\n",intime,ntohs(iph->
tot_len),totalpacket); // time, total length, total packet
        }
  }
  if(iph->protocal==17) //UDP packet
  {
        udph=(struct udphdr*)((char *)iph+(iph->ihl*4)); //get UDP header
        if(ntohs(udph->source) == 6970) // RTP header, port number =
6970(change to your port number)
        {
              totalpacket++;
              printk("Darwin                    \t%ld\t%d\t%d\n",intime,ntohs(iph->
tot_len),totalpacket); // time, total length, total packet
        }
        else if(ntohs(udph->source)==7078) //linphne audio
        {
              totalpacket++;
              printk("Socket_i\t%ld\t%d\t%d\n",intime,ntohs(iph->
tot_len),totalpacket); // time, total length, total packet
        }
  }
  return 0;
}
```

Output.queue.c

```
int piccount = 0;
```

```
int count(struct sk_buff* skb)
{
    struct iphdr *iph; //IP header
    struct tcphdr *tcph; //TCP header
    //struct udphdr *udph; //UDP header
    struct timeval tv; //time struct(second & micro second)
    static int totalpacket=0;
    do_gettimeofday(&tv); //get now time
    _kernel_seconds_t outtime=(tv.tv_sec*1000000)+tv.tv_usec; // intime = micro second
    iph = ip_hdr(skb); //get IP header

    if(iph->protocal==6){ //TCP packet
        tcph=(struct tcphdr *)((char *)iph+(iph->ihl*4));
        if(ntohs(tcph->source) == 8888) // TCP header, port number = 8888(change to your port number)
        {
            totalpacket++;
            printk("Socket_i\t%ld\t%d\t%d\n",outtime,ntohs(iph->
tot_len),totalpacket); // time, total length, total packet
        }
    }
    return 0;
}
```

Packet.throughput.awk

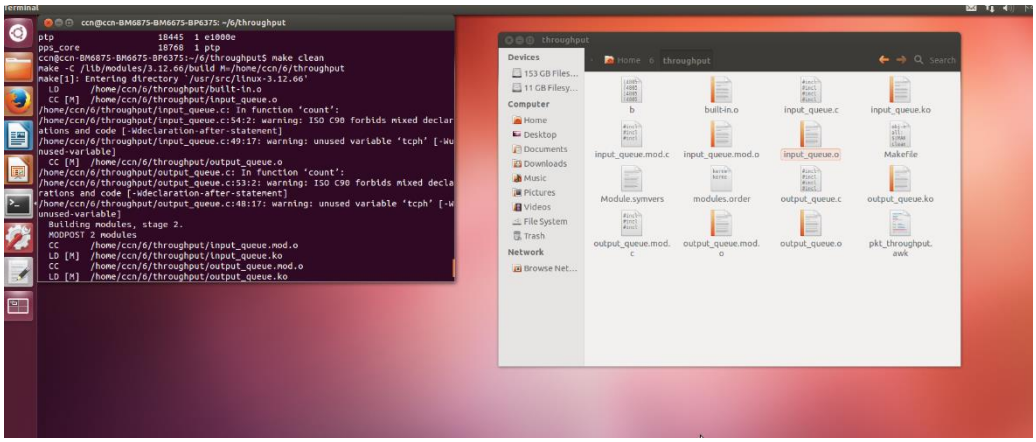
```
BEGIN{
    i=1;
    j=1;
    a=1:
    b=1;
  }
  {
    category = $2;
    t_length = $3;
    time = $4;

    if( category == "Darwin")
    {
        length_d[i] = t_length*8;
        time_d[i] = time;
        i++;
    }
    else if( category == "Asterisk")
```

```
 {
      length_a[j] = t_length*8;
      time_a[j] = time;
      j++;
 }
}

END{
 for(x=1;x<i;x++)
 {
      total_length = total_length + length_d[x];
      total_length_d = total_length;
 }

 for(y=1;y<i;y++)
 {
      total_length = total_length + length_a[y];
      total_length_a = total_length;
 }

 packet_time_d = time_d[i-1]-time_d[1];
 packet_time_a = time_a[j-1]-time_a[1];
 avg_throughput_d = total_length_d / packet_time_d *1000;
 avg_throughput_a = total_length_a / packet_time_a *1000;
 printf("%d KB last %d first %d total %d\n",total_length_d/8/1000,time_d[i-
1],time_d[1],packet_time_d);
 printf("Darwin throughput = %f Kbps\n",avg_throughput_d);
 printf("Darwin throughput = %f KBps\n",avg_throughput_d/8);
 printf("Asterisk throughput = %f Kbps\n",avg_throughput_a);
}
```

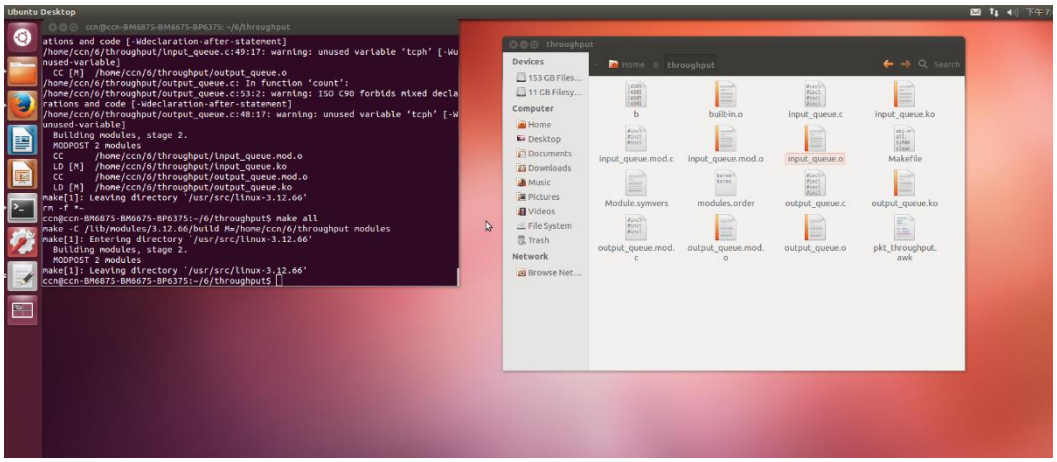
10.2.把寫好的 backdoor program 存入新建的資料夾“throughput”，終端機進入該目錄下面

10.3.make ,把 input_queue, output_queue 的 C 檔 make 成 ko 檔

10.4.make clean



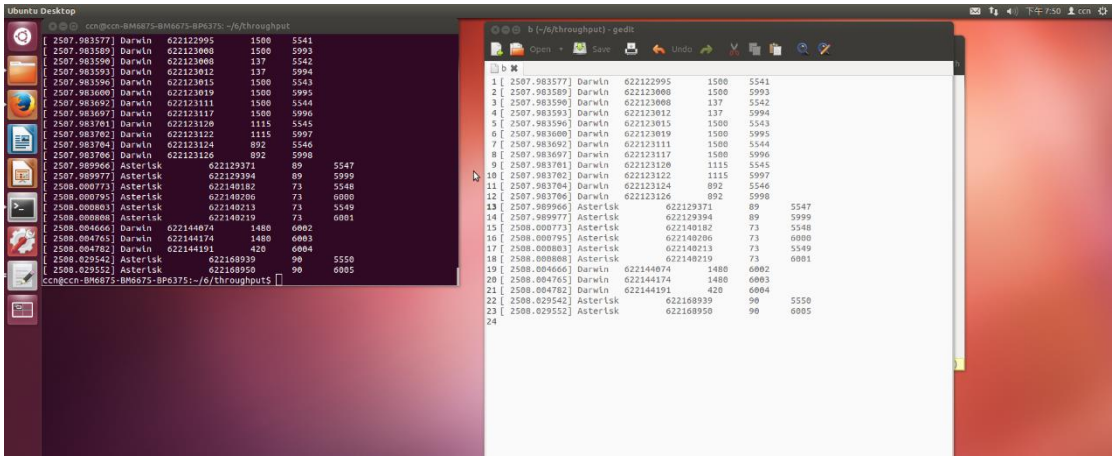
10.5.make all



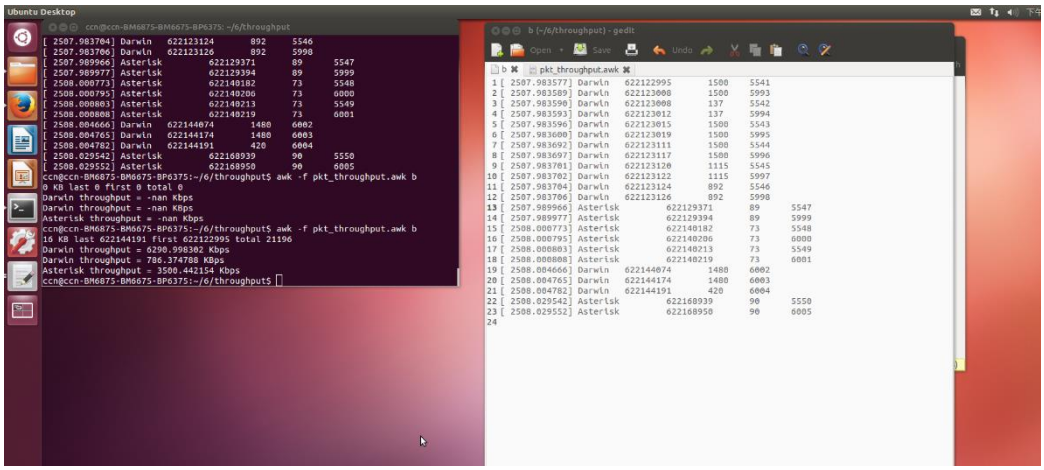
10.6. sudo rmmod input_queue，移除 input_queue.c
10.7. sudo rmmod output_queue，移除 output_queue.c
10.8.sudo insmod input_queue.ko，插入 input_queue.ko
10.9. sudo insmod output_queue.ko，插入 output_queue.ko

10.10.client 端和 server 使用 Darwin 互傳影片

10.11. dmesg,將獲取的詳細資料複製粘貼到 b 檔



10.12. awk –f Delay_time.awk a,計算 delay time



Darwin throughput=6296.998302 kbps
Asterisk throughput=3500.442154kbps

```
[ 2507.983704] Darwin   622123124        892      5546
[ 2507.983706] Darwin   622123126        892      5998
[ 2507.989966] Asterisk          622129371       89       5547
[ 2507.989977] Asterisk          622129394       89       5999
[ 2508.000773] Asterisk          622140182       73       5548
[ 2508.000795] Asterisk          622140206       73       6000
[ 2508.000803] Asterisk          622140213       73       5549
[ 2508.000808] Asterisk          622140219       73       6001
[ 2508.004666] Darwin   622144074        1480     6002
[ 2508.004765] Darwin   622144174        1480     6003
[ 2508.004782] Darwin   622144191        420      6004
[ 2508.029542] Asterisk          622168939       90       5550
[ 2508.029552] Asterisk          622168950       90       6005
```

解釋：因爲 avg_throughput_ = total_length_ / packet_time_ *1000，而根據我們傳輸過程中產生的數據，第 3 列是 total time,第 4 列是 total length,第 5 列是 total packet,可以看出第 3 列和第 5 列的數字大小都差別不是很大，但是第 4 列 total length，Darwin 比 Asterisk 大一個數量級，所以在 total time 差別不太的情況下，Darwin 的 average throughput 比 Asterisk 的大很多。因爲 Darwin 是 video server, video 每秒要播幾十張畫面，video 當然比 audio 大很多。

11.Count IP Packets for Different Size of Text Message

11.1.write down backdoor program

Input_queue.c

```
int count(struct sk_buff* skb)
{
    struct iphdr *iph; //IP header
    struct tcphdr *tcph; //TCP header
    struct udphdr *udph; //UDP header
    struct timeval tv; //time struct(second & micro second)
    static int totalpacket=0;
```

```
    do_gettimeofday(&tv); //get now time
    _kernel_seconds_t intime=(tv.tv_sec*1000000)+tv.tv_usec; // intime = micro
second
    iph = ip_hdr(skb); //get IP header

    if(iph->protocal==6){ //TCP packet
        tcph=(struct tcphdr *)((char *)iph+(iph->ihl*4));
        if(ntohs(tcph->source) == 8888) // TCP header, port number =
8888(change to your port number)
        {
            totalpacket++;
            printk("Socket_i\t%ld\t%d\t%d\n",intime,ntohs(iph->
tot_len),totalpacket); // time, total length, total packet
        }


    }
    return 0;
}
```

Output_queue.c

```
    int count(struct sk_buff* skb)
    {
     struct iphdr *iph; //IP header
     struct tcphdr *tcph; //TCP header
     //struct udphdr *udph; //UDP header
     struct timeval tv; //time struct(second & micro second)
     static int totalpacket=0;
     do_gettimeofday(&tv); //get now time
     _kernel_seconds_t outtime=(tv.tv_sec*1000000)+tv.tv_usec; // intime =
    micro second
     iph = ip_hdr(skb); //get IP header

     if(iph->protocal==6){ //TCP packet
        tcph=(struct tcphdr *)((char *)iph+(iph->ihl*4));
        if(ntohs(tcph->source) == 8888) // TCP header, port number =
    8888(change to your port number)
        {
            totalpacket++;
            printk("Socket_i\t%ld\t%d\t%d\n",outtime,ntohs(iph->
    tot_len),totalpacket); // time, total length, total packet
        }


     }
```

```
  return 0;
}
```

Count_packet.awk

```
BEGIN{
  packets=0;
  i=1;
}
{
  category = $2;
  size = $4;

  if( category == "Socket_i")
  {
        packet_size[i] = size;
        c[i] = category;
        i++;
        packets++;
  }
  else if( category == "Socket_o")
  {
        packet_size[i] = size;
        c[i] = category;
        i++;
        packets++;
  }
}
END{
  for(j=1;j<i;j++)
  {
        total_size = total_size + packet_size[j];
        if(c[j]=="Socket_i")
              printf("Input Packet size %d = %d\n",j,packet_size[j]);
        if(c[j]=="Socket_o")
              printf("Output Packet size %d = %d\n",j,packet_size[j]);
  }

  printf("Total packets = %d\n",packets);
  printf("Total packet size = %d\n",total_size);
}
```

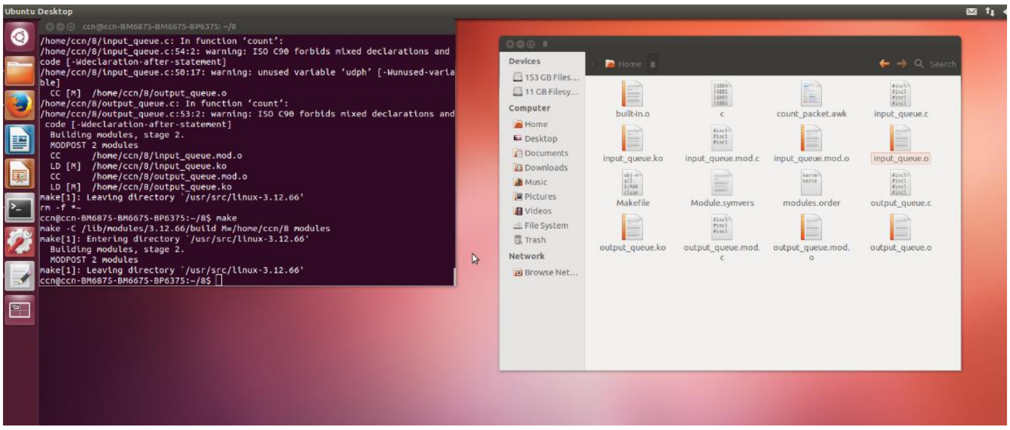
11.2.把寫好的 backdoor program 存入新建的資料夾，終端機進入該目錄下面

11.3.make ,把 input_queue, output_queue 的 C 檔 make 成 ko 檔

11.4.make clean

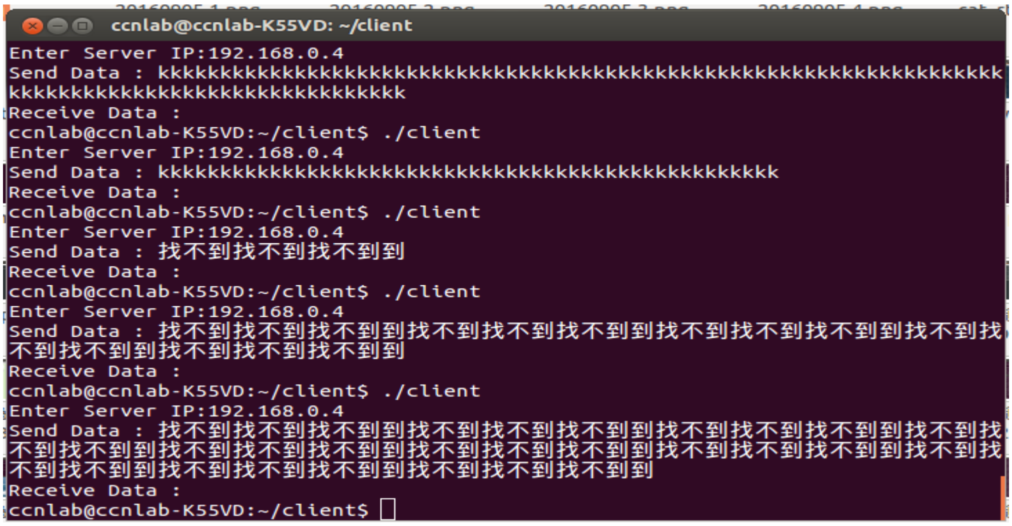
11.5.make all

11.6. sudo rmmod input_queue， 移除 input_queue.c

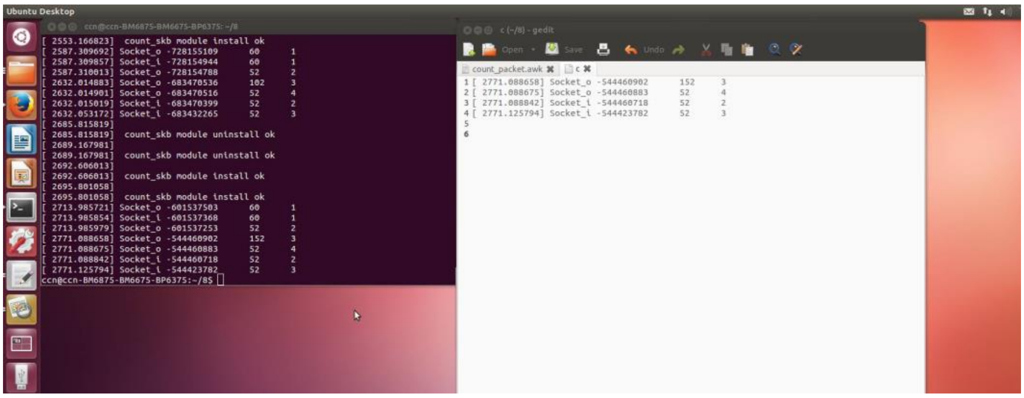


11.7. sudo rmmod output_queue， 移除 output_queue.c
11.8.sudo insmod input_queue.ko, 插入 input_queue.ko
11.9. sudo insmod output_queue.ko, 插入 output_queue.ko

10.10. server 端分別發送 10，50，100 個英文/中文字符

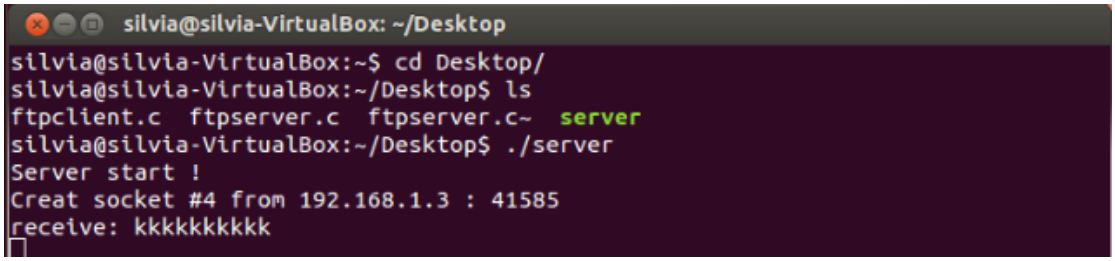


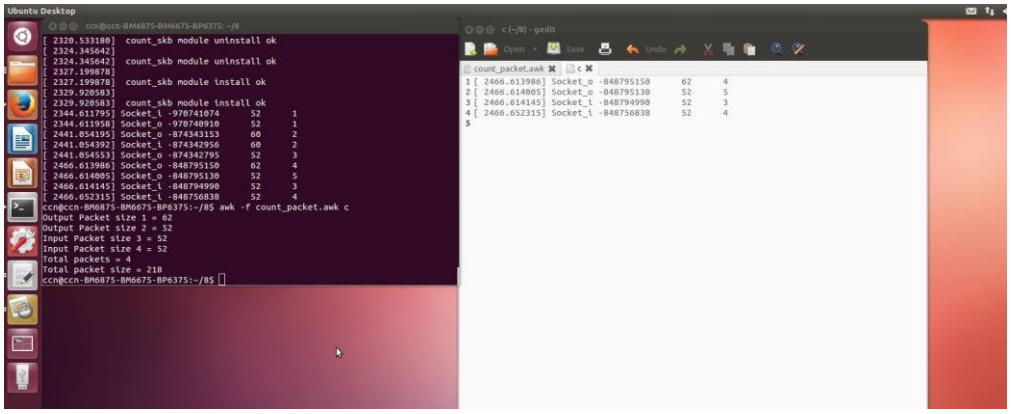
10.11. dmesg,將獲取的詳細資料複製粘貼到 c 檔



11.12.awk –f packet_count.awk c 分別計算 IP packet

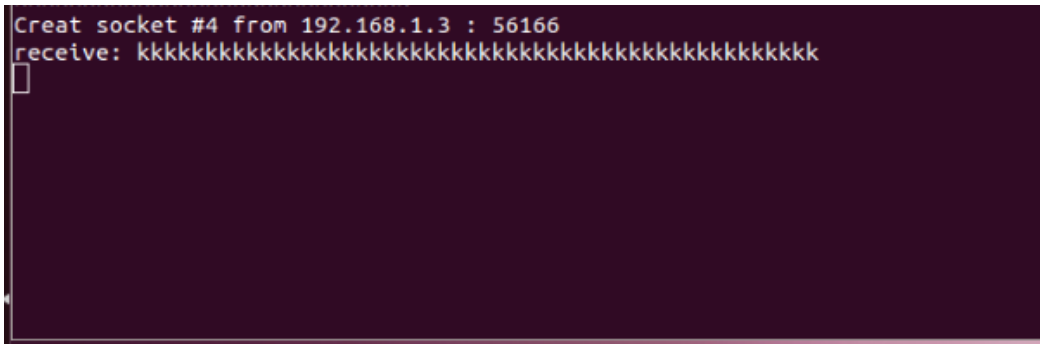
11.13.server 端接收到 10 個英文

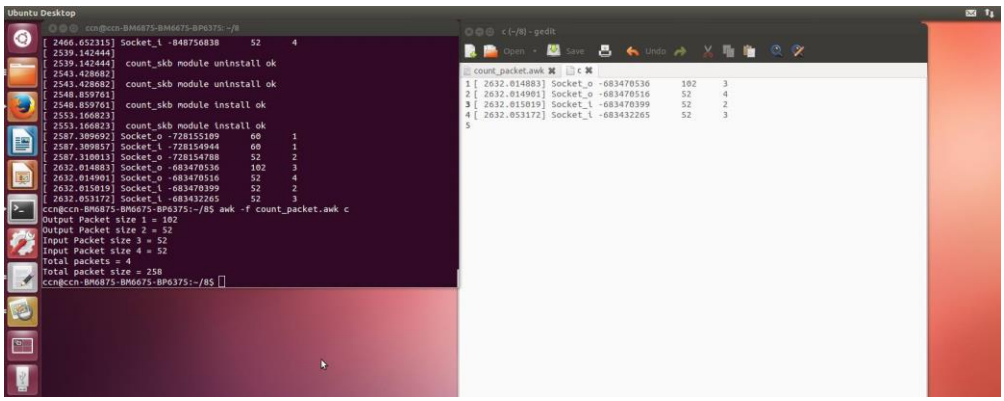




Total packets=4
Total packet size=218

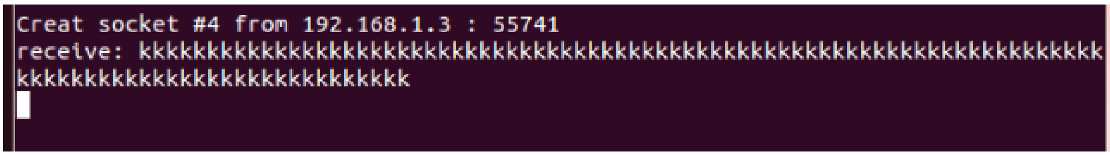
10.5.server 端接收到 50 個英文

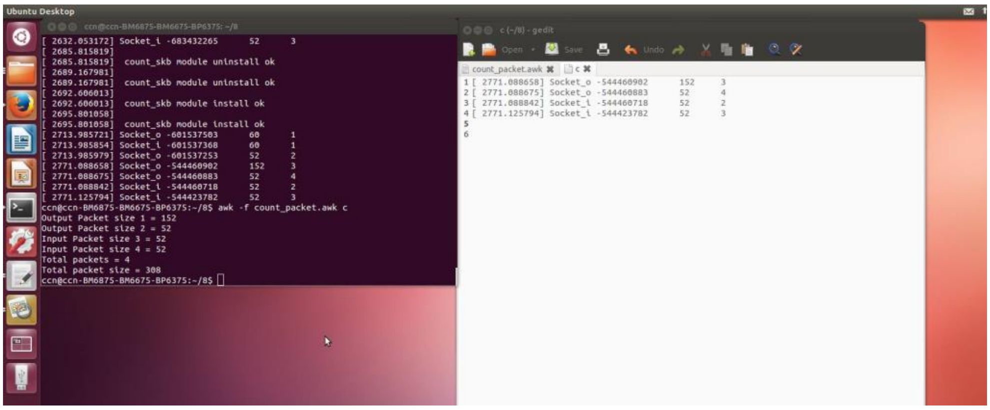




Total packets=4
Total packet size=258

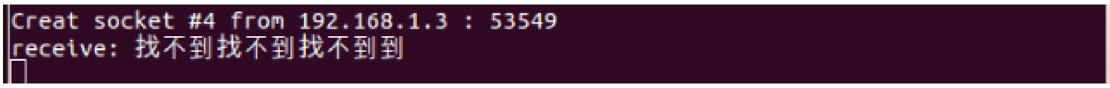
10.6.server 端接收到 100 個英文

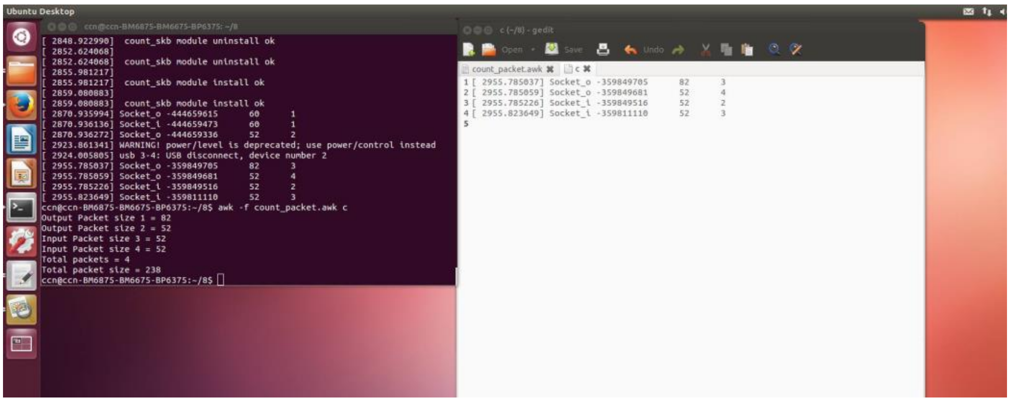




Total packets=4
Total packet size=308

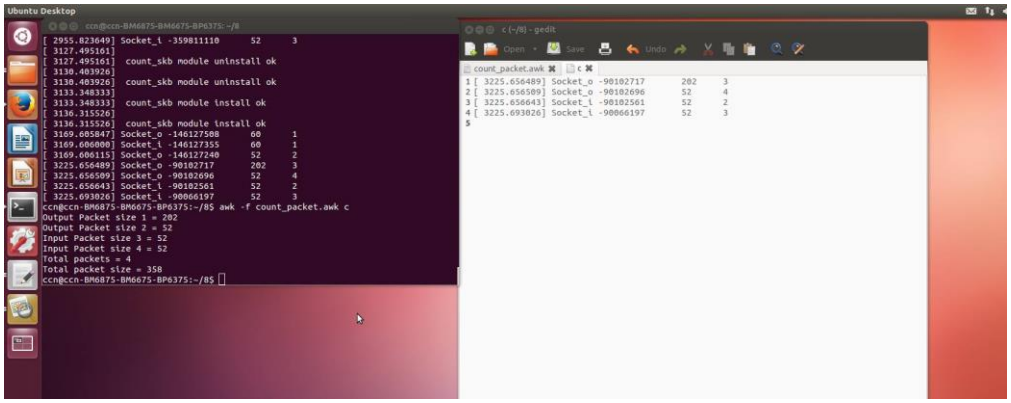
10.7.server 端接收到 10 個中文





Total packets=4
Total packet size=238

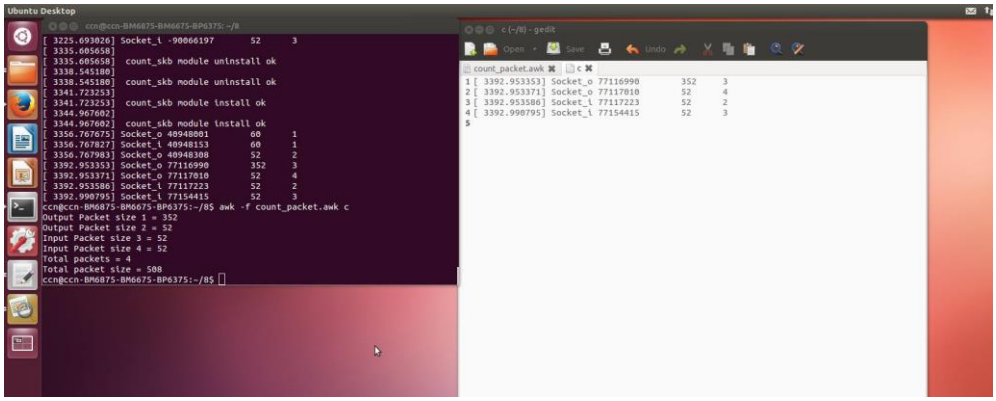
10.8.server 端接收到 50 個中文



Total packets=4

Total packet size=358

10.9.server 端接收到 100 個中文



Total packets=4

Total packet size=508

通過以上計算我們可以發現，中英文字符的 IP packets 个数相同，相同长度的中/英文 message 的 packet size 不同，不同長度的中文（英文）message 的 packet size 也不同，因为根据 ASCII 编码，一个中文字符是 2bytes，一个英文字符是 1byte.