EL5373

INTERNET ARCHITECTURE AND PROTOCOLS

Runze Dong

N10264442

rd1711@nyu.edu

Workstation: APAH Othello_I

MAC: f8:0f:41:c4:7f:aa

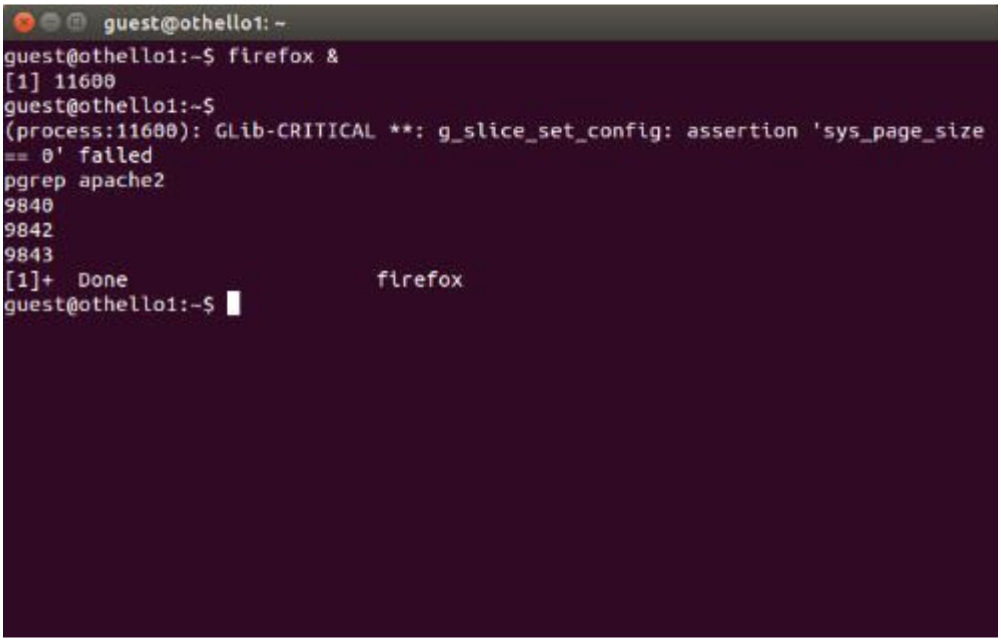
Lab Report 8

Exercise 1

How many http processes were started? Which one was the master server and which ones were the child servers?

3 processes were running. PID 9840, 9842, 9843

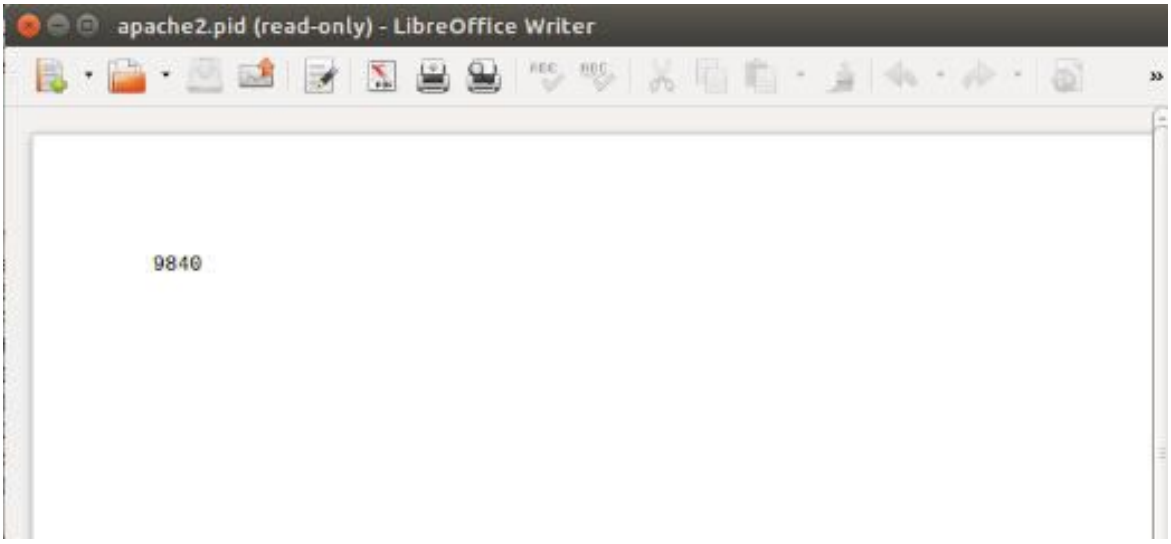
9840 is master server. 9842 and 9843 are child servers.



guest@othello1: ~

```
guest@othello1:~$ firefox &
[1] 11600
guest@othello1:~$
(process:11600): GLib-CRITICAL **: g_slice_set_config: assertion 'sys_page_size
== 0' failed
pgrep apache2
9840
9842
9843
[1]+  Done                    firefox
guest@othello1:~$
```

Justify your answer using the /etc/apache2/envvars file.



apache2.pid (read-only) - LibreOffice Writer

```
9840
```

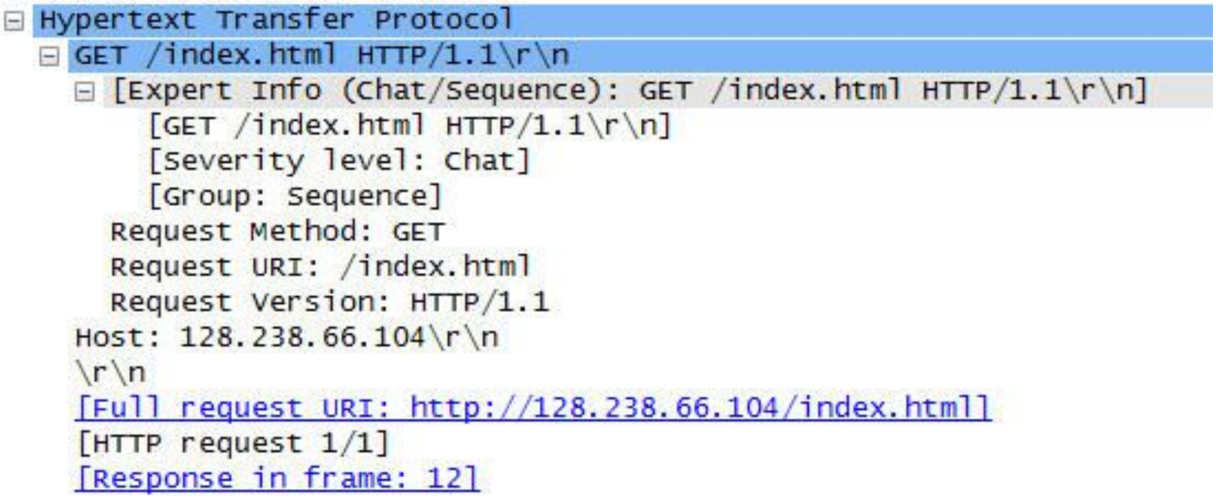
What is the purpose of initiating multiple http processes?

There are mainly two advantages. Firstly when one child process is crashed, others will not be affected. Secondly, more child process can response more client request simultaneously.

Exercise 2

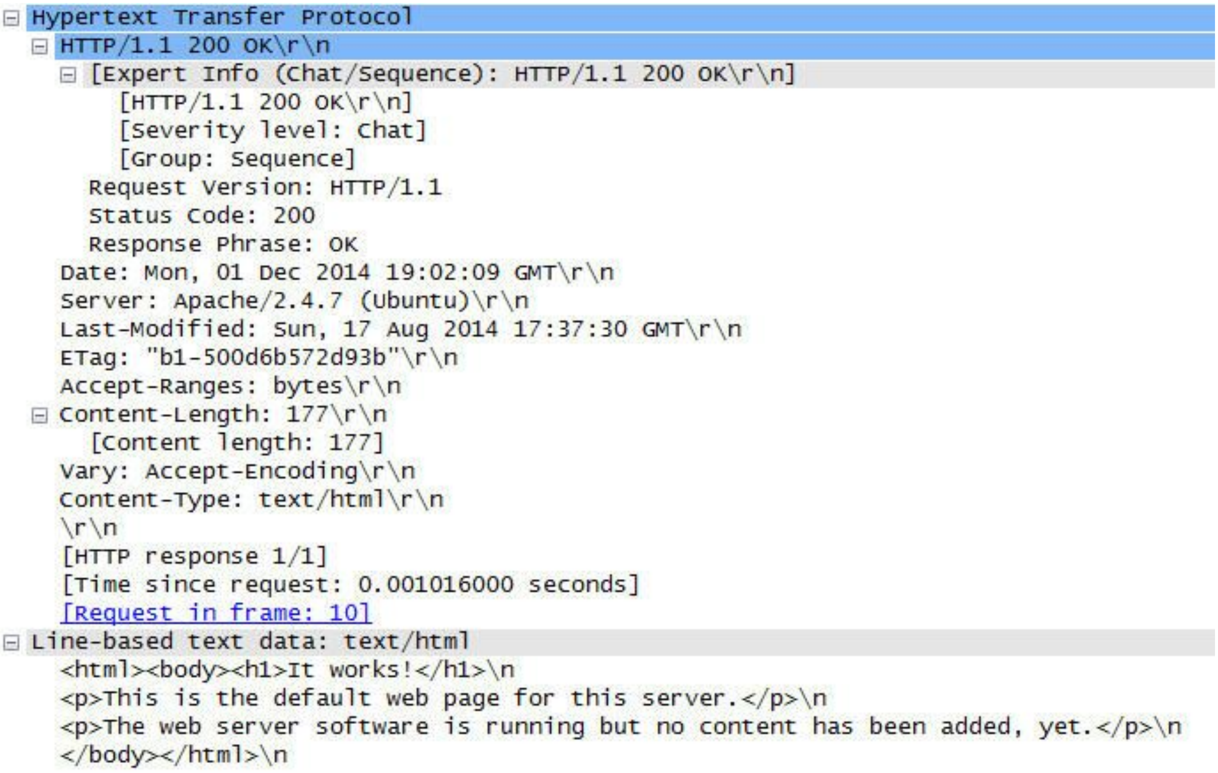
Submit the HTTP request and response, including the start-lines and all the headers.

HTTP client request



```
⊟ Hypertext Transfer Protocol
  ⊟ GET /index.html HTTP/1.1\r\n
    ⊟ [Expert Info (Chat/Sequence): GET /index.html HTTP/1.1\r\n]
        [GET /index.html HTTP/1.1\r\n]
        [Severity level: Chat]
        [Group: Sequence]
      Request Method: GET
      Request URI: /index.html
      Request Version: HTTP/1.1
    Host: 128.238.66.104\r\n
    \r\n
    [Full request URI: http://128.238.66.104/index.html]
    [HTTP request 1/1]
    [Response in frame: 12]
```

HTTP server response



```
⊟ Hypertext Transfer Protocol
  ⊟ HTTP/1.1 200 OK\r\n
    ⊟ [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
        [HTTP/1.1 200 OK\r\n]
        [Severity level: Chat]
        [Group: Sequence]
      Request Version: HTTP/1.1
      Status Code: 200
      Response Phrase: OK
    Date: Mon, 01 Dec 2014 19:02:09 GMT\r\n
    Server: Apache/2.4.7 (Ubuntu)\r\n
    Last-Modified: Sun, 17 Aug 2014 17:37:30 GMT\r\n
    ETag: "b1-500d6b572d93b"\r\n
    Accept-Ranges: bytes\r\n
  ⊟ Content-Length: 177\r\n
      [Content length: 177]
    Vary: Accept-Encoding\r\n
    Content-Type: text/html\r\n
    \r\n
    [HTTP response 1/1]
    [Time since request: 0.001016000 seconds]
    [Request in frame: 10]
⊟ Line-based text data: text/html
    <html><body><h1>It works!</h1>\n
    <p>This is the default web page for this server.</p>\n
    <p>The web server software is running but no content has been added, yet.</p>\n
    </body></html>\n
```

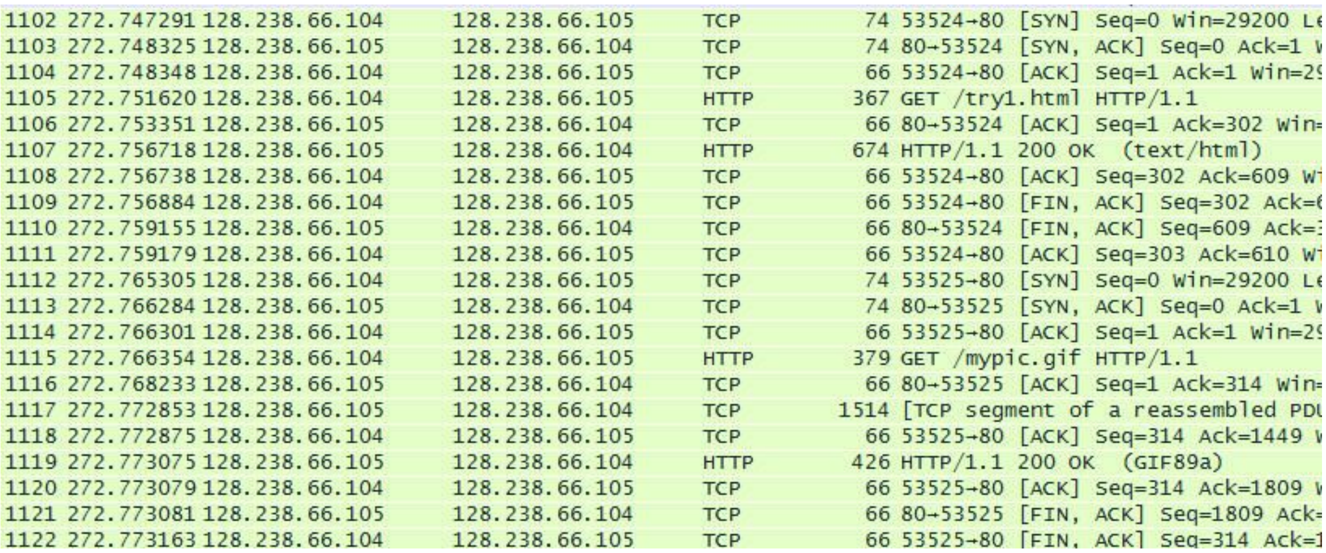
Exercise 3

When you browsed the try1.html file for the first time, how many HTTP requests were sent? Which files were requested? How many TCP connections were used?
3 Http requests. Request for 1). try1.html 2). mypic.gif 3). Favicon.ico
3 TCP connections are used.

Answer the above questions for when you browsed the try1.html file for the second time.
2 Http requests. Request for 1). try1.html 2). mypic.gif
Only one TCP connections is used.

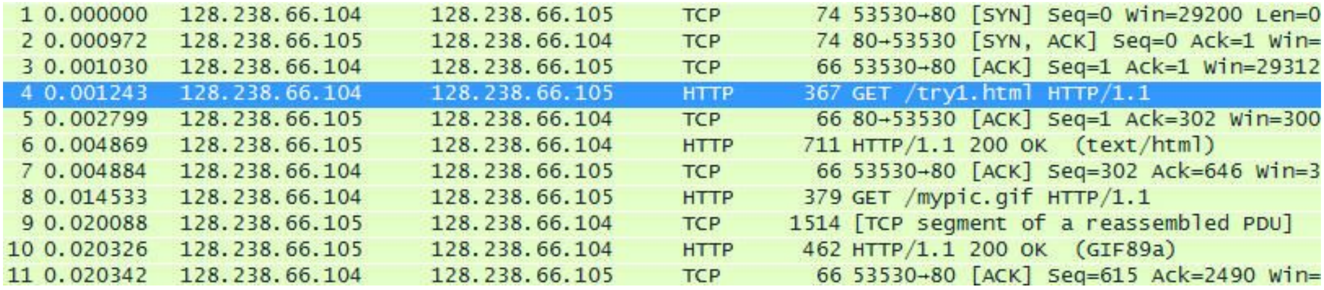
What is the purpose of using persistent connections?
In http/1.0, every http request even for embedded items in html file requires to establish individual TCP connections to transmission. This will waste network resource, increase server/client works and cause additional delay in transmission. So in http/1.1, persistent connections is adopted to solve this inefficiency. It allows embedded items to send through the first http request, which was established for original html file.

First Time try.



```
1102 272.747291 128.238.66.104    128.238.66.105    TCP      74 53524→80 [SYN] Seq=0 Win=29200 L
1103 272.748325 128.238.66.105    128.238.66.104    TCP      74 80→53524 [SYN, ACK] Seq=0 Ack=1
1104 272.748348 128.238.66.104    128.238.66.105    TCP      66 53524→80 [ACK] Seq=1 Ack=1 Win=2
1105 272.751620 128.238.66.104    128.238.66.105    HTTP    367 GET /try1.html HTTP/1.1
1106 272.753351 128.238.66.105    128.238.66.104    TCP      66 80→53524 [ACK] Seq=1 Ack=302 Win
1107 272.756718 128.238.66.105    128.238.66.104    HTTP    674 HTTP/1.1 200 OK  (text/html)
1108 272.756738 128.238.66.104    128.238.66.105    TCP      66 53524→80 [ACK] Seq=302 Ack=609 W
1109 272.756884 128.238.66.104    128.238.66.105    TCP      66 53524→80 [FIN, ACK] Seq=302 Ack=
1110 272.759155 128.238.66.105    128.238.66.104    TCP      66 80→53524 [FIN, ACK] Seq=609 Ack=
1111 272.759179 128.238.66.104    128.238.66.105    TCP      66 53524→80 [ACK] Seq=303 Ack=610 W
1112 272.765305 128.238.66.104    128.238.66.105    TCP      74 53525→80 [SYN] Seq=0 Win=29200 L
1113 272.766284 128.238.66.105    128.238.66.104    TCP      74 80→53525 [SYN, ACK] Seq=0 Ack=1
1114 272.766301 128.238.66.104    128.238.66.105    TCP      66 53525→80 [ACK] Seq=1 Ack=1 Win=2
1115 272.766354 128.238.66.104    128.238.66.105    HTTP    379 GET /mypic.gif HTTP/1.1
1116 272.768233 128.238.66.105    128.238.66.104    TCP      66 80→53525 [ACK] Seq=1 Ack=314 Win
1117 272.772853 128.238.66.105    128.238.66.104    TCP    1514 [TCP segment of a reassembled PD
1118 272.772875 128.238.66.104    128.238.66.105    TCP      66 53525→80 [ACK] Seq=314 Ack=1449
1119 272.773075 128.238.66.105    128.238.66.104    HTTP    426 HTTP/1.1 200 OK  (GIF89a)
1120 272.773079 128.238.66.104    128.238.66.105    TCP      66 53525→80 [ACK] Seq=314 Ack=1809
1121 272.773081 128.238.66.105    128.238.66.104    TCP      66 80→53525 [FIN, ACK] Seq=1809 Ack
1122 272.773163 128.238.66.104    128.238.66.105    TCP      66 53525→80 [FIN, ACK] Seq=314 Ack=
```

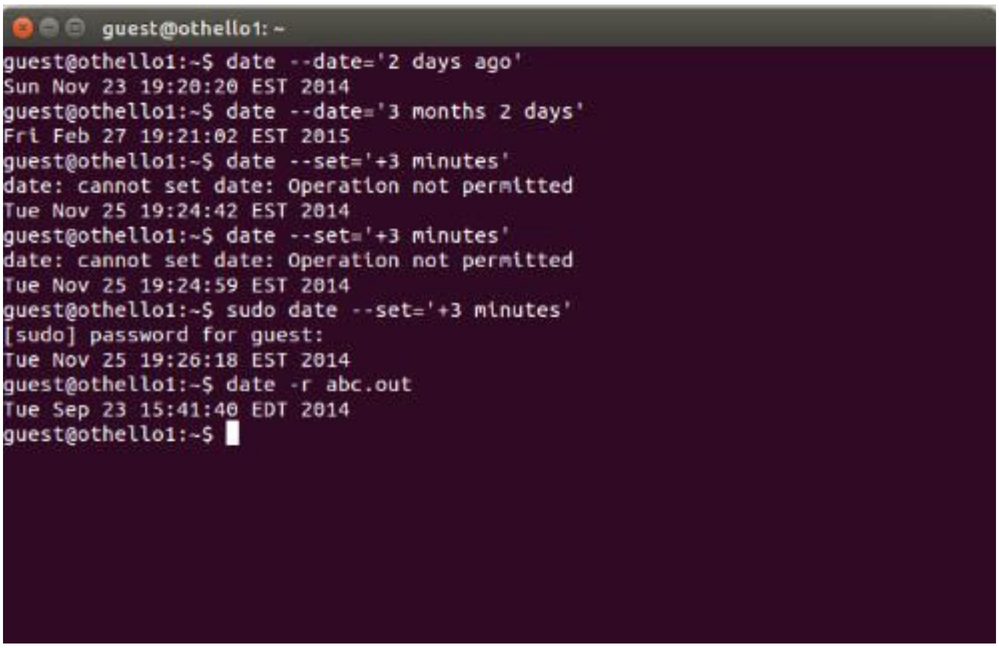
Second time try



```
 1 0.000000  128.238.66.104   128.238.66.105   TCP     74 53530→80 [SYN] Seq=0 Win=29200 Len=0
 2 0.000972  128.238.66.105   128.238.66.104   TCP     74 80→53530 [SYN, ACK] Seq=0 Ack=1 Win=
 3 0.001030  128.238.66.104   128.238.66.105   TCP     66 53530→80 [ACK] Seq=1 Ack=1 Win=29312
 4 0.001243  128.238.66.104   128.238.66.105   HTTP   367 GET /try1.html HTTP/1.1
 5 0.002799  128.238.66.105   128.238.66.104   TCP     66 80→53530 [ACK] Seq=1 Ack=302 Win=300
 6 0.004869  128.238.66.105   128.238.66.104   HTTP   711 HTTP/1.1 200 OK  (text/html)
 7 0.004884  128.238.66.104   128.238.66.105   TCP     66 53530→80 [ACK] Seq=302 Ack=646 Win=3
 8 0.014533  128.238.66.104   128.238.66.105   HTTP   379 GET /mypic.gif HTTP/1.1
 9 0.020088  128.238.66.105   128.238.66.104   TCP   1514 [TCP segment of a reassembled PDU]
10 0.020326  128.238.66.105   128.238.66.104   HTTP   462 HTTP/1.1 200 OK  (GIF89a)
11 0.020342  128.238.66.104   128.238.66.105   TCP     66 53530→80 [ACK] Seq=615 Ack=2490 Win=
```

Exercise 6

Submit the date outputs you saved. Explain the use of the commands.



```
guest@othello1:~$ date --date='2 days ago'
Sun Nov 23 19:20:20 EST 2014
guest@othello1:~$ date --date='3 months 2 days'
Fri Feb 27 19:21:02 EST 2015
guest@othello1:~$ date --set='+3 minutes'
date: cannot set date: Operation not permitted
Tue Nov 25 19:24:42 EST 2014
guest@othello1:~$ date --set='+3 minutes'
date: cannot set date: Operation not permitted
Tue Nov 25 19:24:59 EST 2014
guest@othello1:~$ sudo date --set='+3 minutes'
[sudo] password for guest:
Tue Nov 25 19:26:18 EST 2014
guest@othello1:~$ date -r abc.out
Tue Sep 23 15:41:40 EDT 2014
guest@othello1:~$ 
```

date --date=’ STRING’ means display time described by STRING, not 'now'.
In this lab, date --date=’ 2 days ago’ shows the date of 2 days ago from now.
date --date=’3 months 2 days’ shows the date of 3 months 2 days later from now.

date --set=’STRING’ means set time described by STRING.
In this lab, data --set=’+3 minutes’ means plus 3 minutes on the current time.

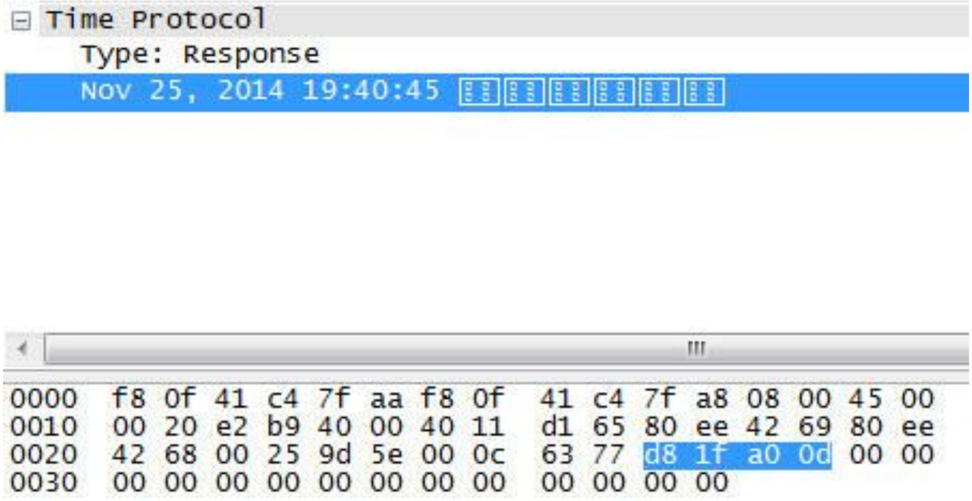
date --r means display the last modification time of FILE
In this lab, date -r abc.out show the last modification time of abc.out.

Exercise 7

What port numbers were used by the remote machine?
Port 37

What port numbers were used by the local host?
Port 33171

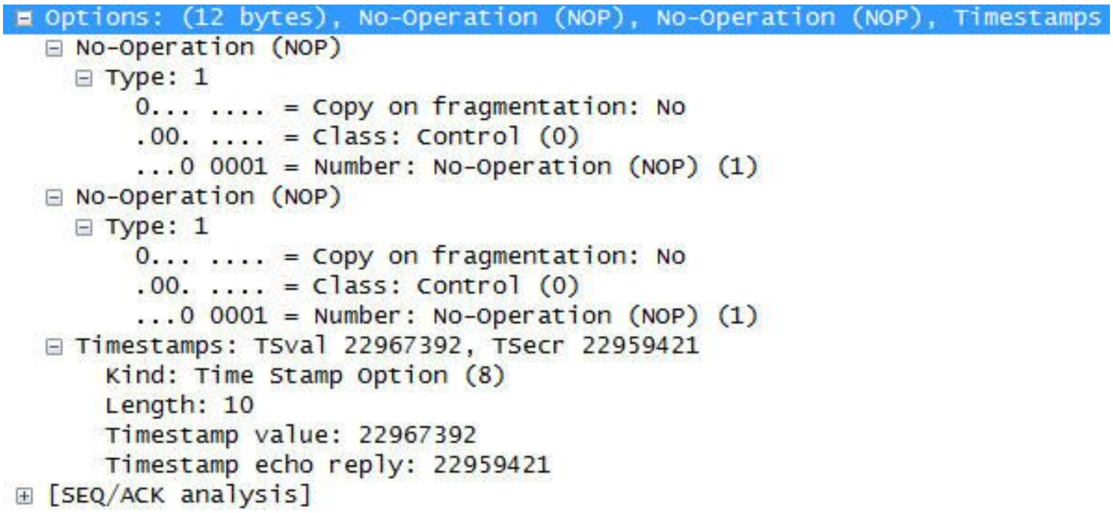
How many bytes of data were returned by the remote time server, both in the UDP case and in the TCP case?
4 bytes data were returned in both cases.



```
⊟ Time Protocol
    Type: Response
    Nov 25, 2014 19:40:45 □□□□□□
```

```
0000  f8 0f 41 c4 7f aa f8 0f  41 c4 7f a8 08 00 45 00
0010  00 20 e2 b9 40 00 40 11  d1 65 80 ee 42 69 80 ee
0020  42 68 00 25 9d 5e 00 0c  63 77 d8 1f a0 0d 00 00
0030  00 00 00 00 00 00 00 00  00 00 00 00
```

What TCP header options were used?
Timestamps(8) option is used.



```
⊟ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ⊟ No-Operation (NOP)
    ⊟ Type: 1
        0... .... = Copy on fragmentation: No
        .00. .... = Class: Control (0)
        ...0 0001 = Number: No-Operation (NOP) (1)
  ⊟ No-Operation (NOP)
    ⊟ Type: 1
        0... .... = Copy on fragmentation: No
        .00. .... = Class: Control (0)
        ...0 0001 = Number: No-Operation (NOP) (1)
  ⊟ Timestamps: TSval 22967392, TSecr 22959421
      Kind: Time Stamp Option (8)
      Length: 10
      Timestamp value: 22967392
      Timestamp echo reply: 22959421
⊞ [SEQ/ACK analysis]
```