PROJECT REPORT

ON

CLIENT-SERVER COMMUNICATION USING ANDROID

SUBMITTED BY: Networking 1 Group

VENKATESH AVULA

SASHIDHAR EVURU

Contents

1. PROJECT DESCRIPTION:

The project involves the TCP communication between the client and the server, client being android mobile device and server being the laptop. Laptop acts as a server by running a server program continuously whereas the mobile acts as a client with an Android console, establishes the connection with the client only when the file exchange is required. This approach even works for communication between two laptops/work stations where one system acts as a client and other as a server.

The project has been implemented using TCP protocol because of the following reasons:

1. TCP provides for the reliable information exchange between the end systems that wishes to communicate. As the file transfer in the project involves the text and image files, TCP ensures that all data grams sent from one end are received at the other end. Hence the quality of the file being transferred is preserved.
2. The project involves communication between the client and server to ensure the files at the both ends are of recent copies. The server checks the timestamp values of the client and server copies of same file and exchanges this information with the client before the actual file exchange occurs. Also, once the client sends/receives the entire copy of file, it sends the "Bye" message to the server. As TCP establishes a connection before the communication (actual file transfer), it is easy and efficient to exchange such back and forth messages between the client and the server.

2. IMPLEMENTATION PROCEDURE:

The server has to be started first and a port number has to be assigned to it. The client application is deployed as an Android application in the mobile device. We have implemented the following scenarios using the Android and Network programming API's for TCP connection establishment and communication.

2.1 Initial system set up:

1. Set the server location to be any directory inside C: drive. We have used C:\Server as the directory in which all server files are available in the laptop.
2. All the files at the client side should be stored in the download directory of the mobile device. We have created a separate folder named MsgFolder folder inside the Download directory to store the files at the client side. Mobile Downloads folder location is obtained using Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOWNLOADS) function provided by Android library.

2.2 Scenarios implemented:

2.2.1 Scenario 1:

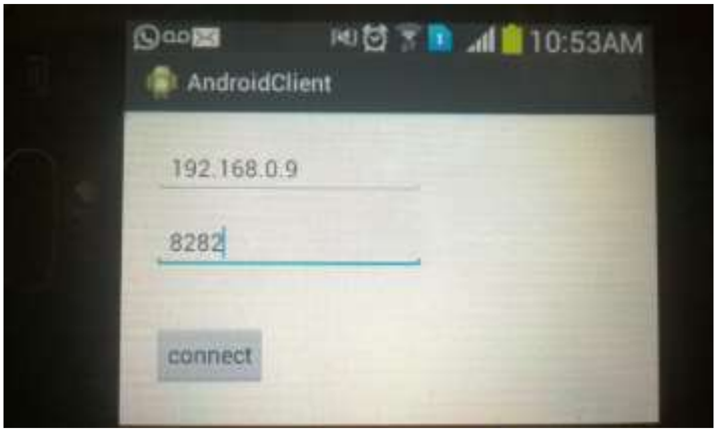
Server contains a file and client wishes to download the recent copy of file from the server. File already exists in the client but the server's copy of file has the recently updated changes. The following steps are involved.

1. As the communication is through TCP, connection establishment occurs before the actual file transfer. Client (mobile device) initiates the connection establishment by entering the IP address and port number of the server (laptop) in the android console.
2. Once the connection has been established, the client enters the file name into the console and checks whether there is a file with the same name and extension in its directory. As the client contains this file, it sets the timestamp value of the file to be the last modified date of the file.
3. Once the 'sync' button is pressed in Android console, the client sends the file name and timestamp of the file it needs to download from the server and starts waiting for the server response.
4. On the other side, once the server receives the file name and time stamp value from the client, the server checks the time stamp values of its own copy of file and the received timestamp.
5. As the server contains the recent file, the time stamp value of the server copy is greater and hence server starts writing the file to the client's output stream.
6. Once the entire file is received at the client side, the client initiates the connection termination by sending a "Bye" message to the server.
7. The server receives the "Bye" message and terminates the connection by closing the socket and the input, output stream objects created for file transfer.

<u>Demo of the Scenario 1 using Wireshark tool:</u>

Step 1:

Connection establishment is initiated through client (mobile device). The android console at the client (mobile device) is displayed as shown:



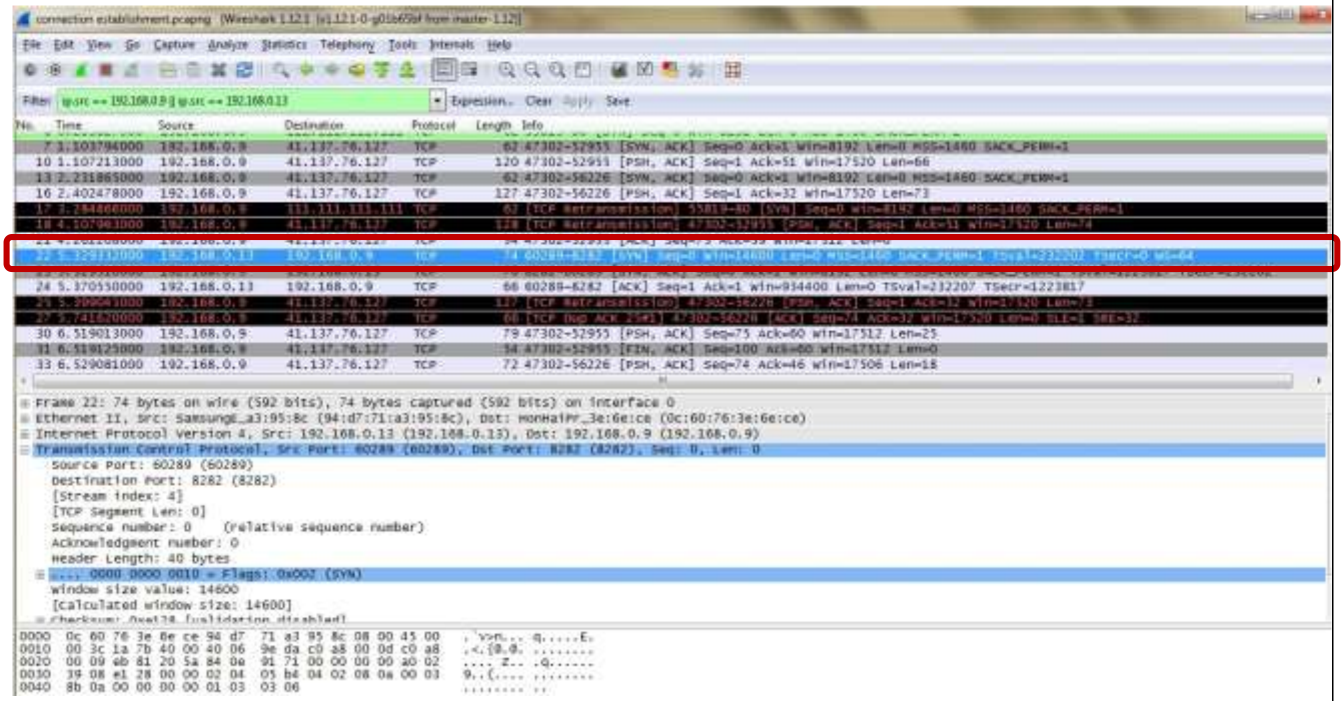
192.168.0.9

8282

In the console, IP address of the laptop and port number in which the server program is running are provided and then once the connect is clicked, the connection request is initiated by the client (mobile device).

Below screenshot shows the corresponding Wireshark capture. The highlighted portion shows the TCP connection request (SYN flag) initiated by mobile device (IP address: 192.168.0.13) to the laptop (IP address: 192.168.0.9) by sending a initial sequence number.

In the protocol stack below, IP layer contains the mobile device IP address as the Source IP address and laptop IP address as the destination IP address. TCP layer contains the source port as the dynamic port 60289 taken by the mobile device whereas the destination port to be the port number 8282 in which the server program is running and to which the client send the connection request.



connection establishment.pcapng [Wireshark 1.12.1 (v1.12.1-0-g01b65bf from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: ip.src == 192.168.0.9 || ip.src == 192.168.0.13
Expression... Clear Apply Save

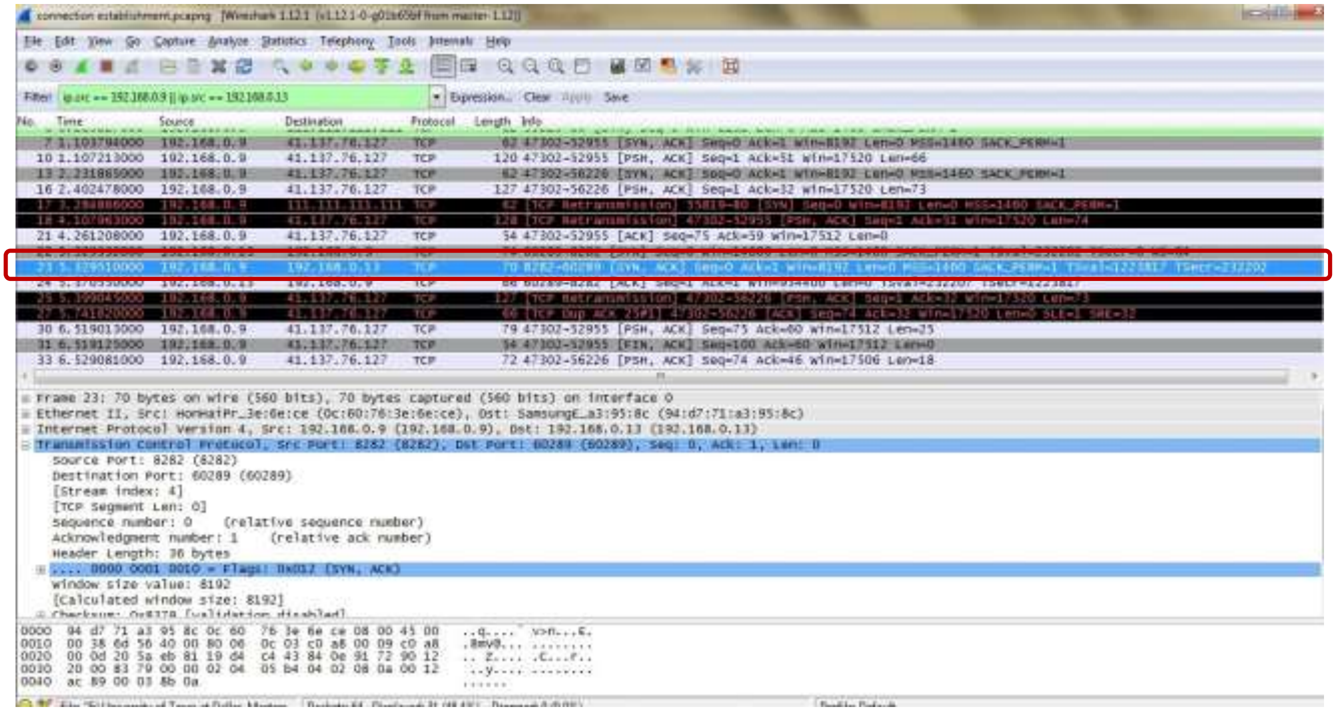
```
No.  Time        Source        Destination     Protocol Length Info
 7 1.103794000 192.168.0.9   41.137.76.127   TCP    62 47302→52955 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 SACK_PERM=1
10 1.107213000 192.168.0.9   41.137.76.127   TCP   120 47302→52955 [PSH, ACK] Seq=1 Ack=51 Win=17520 Len=66
13 2.231865000 192.168.0.9   41.137.76.127   TCP    62 47302→56226 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 SACK_PERM=1
16 2.402478000 192.168.0.9   41.137.76.127   TCP   127 47302→56226 [PSH, ACK] Seq=1 Ack=32 Win=17520 Len=73
17 3.284466000 192.168.0.9   111.111.111.111 TCP    62 [TCP Retransmission] 55819→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 SACK_PERM=1
18 4.107903000 192.168.0.9   41.137.76.127   TCP   128 [TCP Retransmission] 47302→52955 [PSH, ACK] Seq=1 Ack=51 Win=17520 Len=74
22 5.329132000 192.168.0.13  192.168.0.9     TCP    74 60289→8282 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=232202 TSecr=0 WS=64
24 5.370550000 192.168.0.13  192.168.0.9     TCP    66 60289→8282 [ACK] Seq=1 Ack=1 Win=934400 Len=0 TSval=232207 TSecr=1223817
25 5.399043000 192.168.0.9   41.137.76.127   TCP   127 [TCP Retransmission] 47302→56226 [PSH, ACK] Seq=1 Ack=32 Win=17520 Len=73
27 5.741620000 192.168.0.9   41.137.76.127   TCP    66 [TCP Dup ACK 25#1] 47302→56226 [ACK] Seq=74 Ack=32 Win=17520 Len=0 SLE=1 SRE=32
30 6.519013000 192.168.0.9   41.137.76.127   TCP    79 47302→52955 [PSH, ACK] Seq=75 Ack=60 Win=17512 Len=25
31 6.519125000 192.168.0.9   41.137.76.127   TCP    54 47302→52955 [FIN, ACK] Seq=100 Ack=60 Win=17512 Len=0
33 6.529081000 192.168.0.9   41.137.76.127   TCP    72 47302→56226 [PSH, ACK] Seq=74 Ack=46 Win=17506 Len=18

⊞ Frame 22: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
⊞ Ethernet II, Src: SamsungE_a3:95:8c (94:d7:71:a3:95:8c), Dst: HonHaiPr_3e:6e:ce (0c:60:76:3e:6e:ce)
⊞ Internet Protocol Version 4, Src: 192.168.0.13 (192.168.0.13), Dst: 192.168.0.9 (192.168.0.9)
⊟ Transmission Control Protocol, Src Port: 60289 (60289), Dst Port: 8282 (8282), Seq: 0, Len: 0
    Source Port: 60289 (60289)
    Destination Port: 8282 (8282)
    [Stream index: 4]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    Acknowledgment number: 0
    Header Length: 40 bytes
  ⊞ .... 0000 0000 0010 = Flags: 0x002 (SYN)
    Window size value: 14600
    [Calculated window size: 14600]
  ⊞ Checksum: 0xe128 [validation disabled]

0000  0c 60 76 3e 6e ce 94 d7  71 a3 95 8c 08 00 45 00   .`v>n... q.....E.
0010  00 3c 1a 7b 40 00 40 06  9e da c0 a8 00 0d c0 a8   .<.{@.@. ........
0020  00 09 eb 81 20 5a 84 0e  91 71 00 00 00 00 a0 02   .... Z.. .q......
0030  39 08 e1 28 00 00 02 04  05 b4 04 02 08 0a 00 03   9..(.... ........
0040  8b 0a 00 00 00 00 01 03  03 06                     .........  ..
```

Step 2:

Once the connection request is sent by the mobile device, the laptop (server) responds to the SYN request by sending the acknowledgement with ACK flag at source port number (8282) using which server program is running and destination port as the port number (60829) from which the connection request has been received. Also, the protocol stack at IP layer shows the corresponding IP addresses of mobile device and laptop.



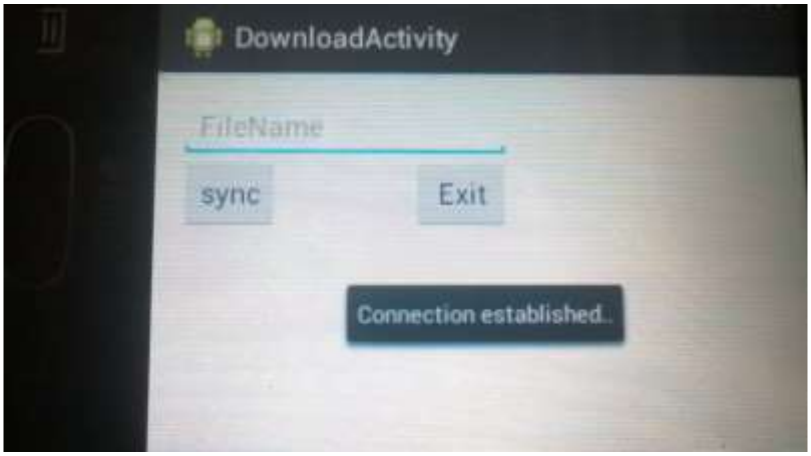
connection establishment.pcapng [Wireshark 1.12.1 (v1.12.1-0-g01b65bf from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: ip.src == 192.168.0.9 || ip.src == 192.168.0.13
Expression... Clear Apply Save

```
No.  Time         Source         Destination      Protocol  Length Info
  7 1.103794000  192.168.0.9    41.137.76.127    TCP        62 47302-52955 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 SACK_PERM=1
 10 1.107213000  192.168.0.9    41.137.76.127    TCP       120 47302-52955 [PSH, ACK] Seq=1 Ack=51 Win=17520 Len=66
 13 2.231885000  192.168.0.9    41.137.76.127    TCP        62 47302-56226 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 SACK_PERM=1
 16 2.402478000  192.168.0.9    41.137.76.127    TCP       127 47302-56226 [PSH, ACK] Seq=1 Ack=32 Win=17520 Len=73
 17 3.284886000  192.168.0.9    111.111.111.111  TCP        62 [TCP Retransmission] 55819-80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 SACK_PERM=1
 18 4.107963000  192.168.0.9    41.137.76.127    TCP       128 [TCP Retransmission] 47302-52955 [PSH, ACK] Seq=1 Ack=51 Win=17520 Len=74
 21 4.261208000  192.168.0.9    41.137.76.127    TCP        54 47302-52955 [ACK] Seq=75 Ack=59 Win=17512 Len=0
 22 5.323332000  192.168.0.13   192.168.0.9      TCP        74 60289-8282 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=232202 TSecr=0 WS=64
 23 5.329510000  192.168.0.9    192.168.0.13     TCP        70 8282-60289 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 SACK_PERM=1 TSval=1223817 TSecr=232202
 24 5.370330000  192.168.0.13   192.168.0.9      TCP        66 60289-8282 [ACK] Seq=1 Ack=1 Win=934400 Len=0 TSval=232207 TSecr=1223817
 25 5.399045000  192.168.0.9    41.137.76.127    TCP       127 [TCP Retransmission] 47302-56226 [PSH, ACK] Seq=1 Ack=32 Win=17520 Len=73
 27 5.741820000  192.168.0.9    41.137.76.127    TCP        66 [TCP Dup ACK 25#1] 47302-56226 [ACK] Seq=74 Ack=32 Win=17520 Len=0 SLE=1 SRE=32
 30 6.519013000  192.168.0.9    41.137.76.127    TCP        79 47302-52955 [PSH, ACK] Seq=75 Ack=60 Win=17512 Len=25
 31 6.519125000  192.168.0.9    41.137.76.127    TCP        54 47302-52955 [FIN, ACK] Seq=100 Ack=60 Win=17512 Len=0
 33 6.529081000  192.168.0.9    41.137.76.127    TCP        72 47302-56226 [PSH, ACK] Seq=74 Ack=46 Win=17506 Len=18
```

```
⊞ Frame 23: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
⊞ Ethernet II, Src: HonHaiPr_3e:6e:ce (0c:60:76:3e:6e:ce), Dst: SamsungE_a3:95:8c (94:d7:71:a3:95:8c)
⊞ Internet Protocol Version 4, Src: 192.168.0.9 (192.168.0.9), Dst: 192.168.0.13 (192.168.0.13)
⊟ Transmission Control Protocol, Src Port: 8282 (8282), Dst Port: 60289 (60289), Seq: 0, Ack: 1, Len: 0
    Source Port: 8282 (8282)
    Destination Port: 60289 (60289)
    [Stream index: 4]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    Acknowledgment number: 1    (relative ack number)
    Header Length: 36 bytes
  ⊞ .... 0000 0001 0010 = Flags: 0x012 (SYN, ACK)
    Window size value: 8192
    [Calculated window size: 8192]
  ⊞ Checksum: 0x8378 [validation disabled]
```

```
0000  94 d7 71 a3 95 8c 0c 60  76 3e 6e ce 08 00 45 00   ..q....` v>n...E.
0010  00 38 6d 56 40 00 80 06  0c 03 c0 a8 00 09 c0 a8   .8mV@... ........
0020  00 0d 20 5a eb 81 19 d4  c4 43 84 0e 91 72 90 12   .. Z.... .C...r..
0030  20 00 83 79 00 00 02 04  05 b4 04 02 08 0a 00 12    ..y.... ........
0040  ac 89 00 03 8b 0a                                  ......
```

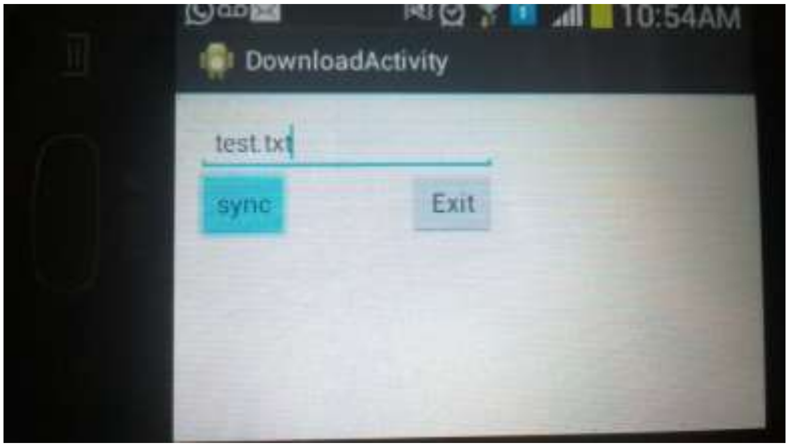
Once the connection is established, the android console gives the 'Connection established' message which suggests the completion of SYN and ACK flags for the connection request between client and the server.



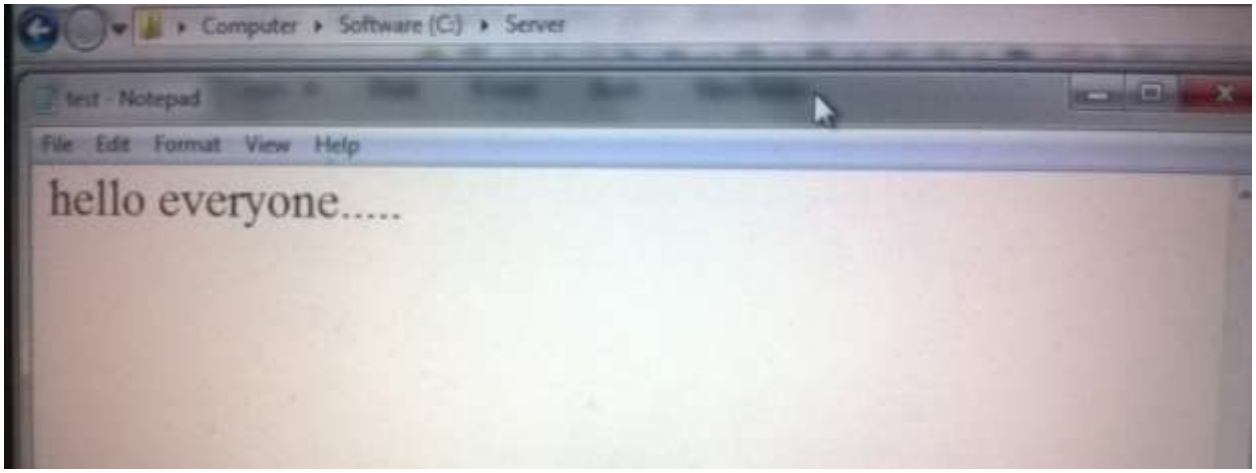
Connection established.

Step 3:

Once the connection has been successfully established, the client (android console) specifies the name of the file to be transferred and clicks on 'sync' button.



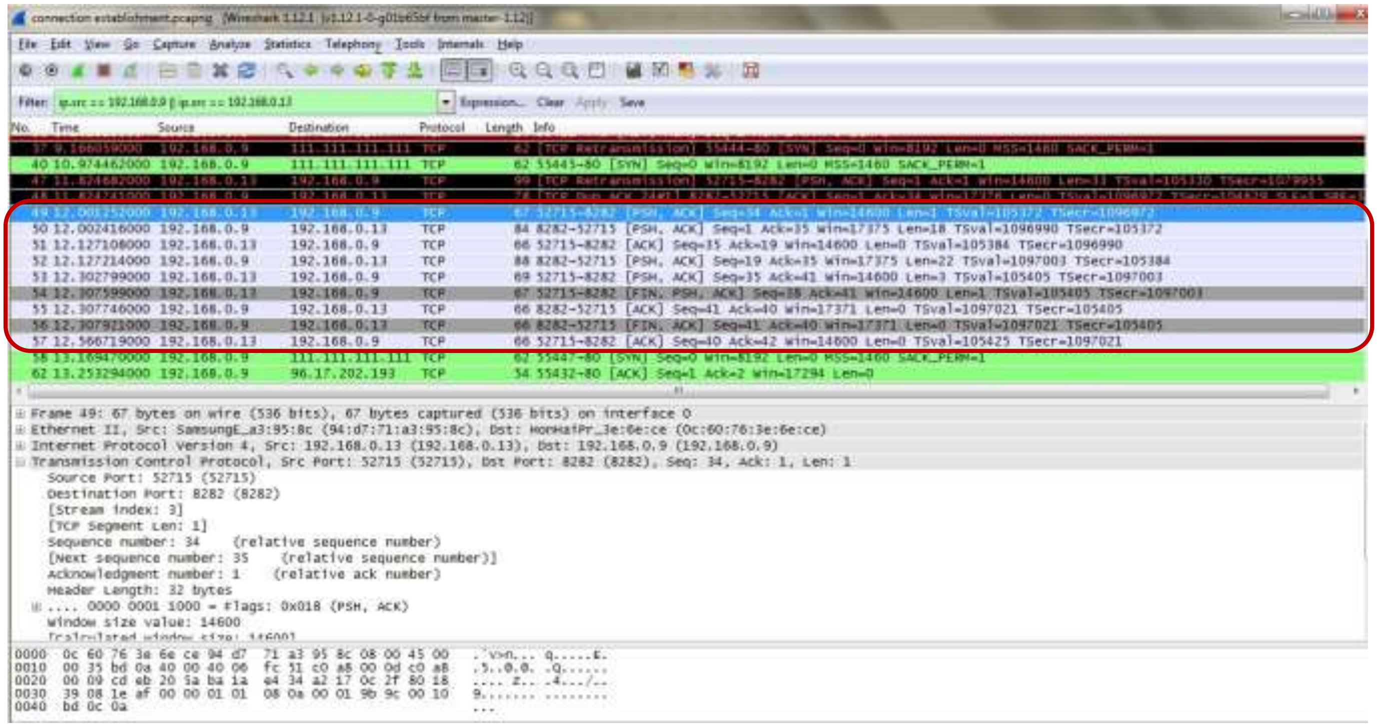
File inside the server directory C:\Server in the laptop:



hello everyone.....

Step 4:

The actual file transfer using the TCP packets is as shown. As in the given scenario, the download happens from server (laptop) to the client (mobile device), file is being sent using TCP packets with PSH flag from laptop (IP address: 192.168.0.9) to mobile device (IP address: 192.168.0.13). For the packets received from one end to other end such as messages between client and server, corresponding acknowledgements are received using the TCP packets with the ACK flag.


connection establishment.pcapng [Wireshark 1.12.1 (v1.12.1-0-g01b65bf from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Expression... Clear Apply Save

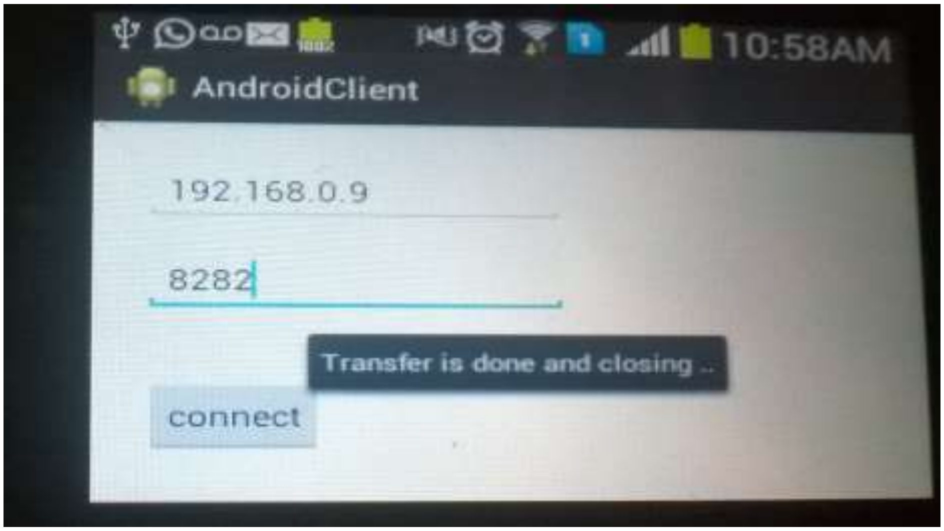
Filter: ip.src == 192.168.0.9 || ip.src == 192.168.0.13

```
No.  Time          Source          Destination      Protocol  Length Info
37 9.166059000   192.168.0.9     111.111.111.111  TCP       62 [TCP Retransmission] 55444→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 SACK_PERM=1
40 10.974462000  192.168.0.9     111.111.111.111  TCP       62 55445→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 SACK_PERM=1
47 11.824682000  192.168.0.13    192.168.0.9      TCP       99 [TCP Retransmission] 52715→8282 [PSH, ACK] Seq=1 Ack=1 Win=14600 Len=33 TSval=105330 TSecr=1079955
48 11.824741000  192.168.0.9     192.168.0.13     TCP       78 [TCP Dup ACK 24#1] 8282→52715 [ACK] Seq=1 Ack=34 Win=17376 Len=0 TSval=1096972 TSecr=104829 SLE=1 SRE=
49 12.001252000  192.168.0.13    192.168.0.9      TCP       67 52715→8282 [PSH, ACK] Seq=34 Ack=1 Win=14600 Len=1 TSval=105372 TSecr=1096972
50 12.002416000  192.168.0.9     192.168.0.13     TCP       84 8282→52715 [PSH, ACK] Seq=1 Ack=35 Win=17375 Len=18 TSval=1096990 TSecr=105372
51 12.127108000  192.168.0.13    192.168.0.9      TCP       66 52715→8282 [ACK] Seq=35 Ack=19 Win=14600 Len=0 TSval=105384 TSecr=1096990
52 12.127214000  192.168.0.9     192.168.0.13     TCP       88 8282→52715 [PSH, ACK] Seq=19 Ack=35 Win=17375 Len=22 TSval=1097003 TSecr=105384
53 12.302799000  192.168.0.13    192.168.0.9      TCP       69 52715→8282 [PSH, ACK] Seq=35 Ack=41 Win=14600 Len=3 TSval=105405 TSecr=1097003
54 12.307599000  192.168.0.13    192.168.0.9      TCP       67 52715→8282 [FIN, PSH, ACK] Seq=38 Ack=41 Win=14600 Len=1 TSval=105405 TSecr=1097003
55 12.307746000  192.168.0.9     192.168.0.13     TCP       66 8282→52715 [ACK] Seq=41 Ack=40 Win=17371 Len=0 TSval=1097021 TSecr=105405
56 12.307921000  192.168.0.9     192.168.0.13     TCP       66 8282→52715 [FIN, ACK] Seq=41 Ack=40 Win=17371 Len=0 TSval=1097021 TSecr=105405
57 12.566719000  192.168.0.13    192.168.0.9      TCP       66 52715→8282 [ACK] Seq=40 Ack=42 Win=14600 Len=0 TSval=105425 TSecr=1097021
58 13.169470000  192.168.0.9     111.111.111.111  TCP       62 55447→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 SACK_PERM=1
62 13.253294000  192.168.0.9     96.17.202.193    TCP       54 55432→80 [ACK] Seq=1 Ack=2 Win=17294 Len=0
```

```
⊞ Frame 49: 67 bytes on wire (536 bits), 67 bytes captured (536 bits) on interface 0
⊞ Ethernet II, Src: SamsungE_a3:95:8c (94:d7:71:a3:95:8c), Dst: HonHaiPr_3e:6e:ce (0c:60:76:3e:6e:ce)
⊞ Internet Protocol Version 4, Src: 192.168.0.13 (192.168.0.13), Dst: 192.168.0.9 (192.168.0.9)
⊟ Transmission Control Protocol, Src Port: 52715 (52715), Dst Port: 8282 (8282), Seq: 34, Ack: 1, Len: 1
    Source Port: 52715 (52715)
    Destination Port: 8282 (8282)
    [Stream index: 3]
    [TCP Segment Len: 1]
    Sequence number: 34    (relative sequence number)
    [Next sequence number: 35    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Header Length: 32 bytes
  ⊞ .... 0000 0001 1000 = Flags: 0x018 (PSH, ACK)
    Window size value: 14600
    [Calculated window size: 14600]
```

```
0000  0c 60 76 3e 6e ce 94 d7  71 a3 95 8c 08 00 45 00   .`v>n... q.....E.
0010  00 35 bd 0a 40 00 40 06  fc 51 c0 a8 00 0d c0 a8   .5..@.@. .Q......
0020  00 09 cd eb 20 5a ba 1a  e4 34 a2 17 0c 2f 80 18   .... Z.. .4.../..
0030  39 08 1e af 00 00 01 01  08 0a 00 01 9b 9c 00 10   9....... ........
0040  bd 0c 0a                                           ...
```

Message is displayed in the Android console once the file transferred is completed as shown:

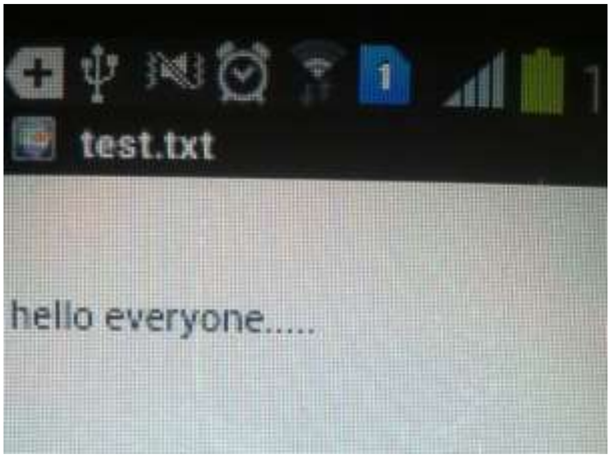


192.168.0.9

8282

Transfer is done and closing ..

The file present at the client side (mobile device) after the completion of download from the server.



test.txt

hello everyone.....

2.2.2 Scenario 2:

Server contains a file and client wishes to download the file. File already exists in the server. Client does not contain this file. The following steps are involved.

1. As the communication is through TCP, connection establishment occurs before the actual file transfer. Client (mobile device) initiates the connection establishment by entering the IP address and port number of the server (laptop) in the android console.
2. Once the connection has been established, the client enters the file name into the console and checks whether there is a file with the same name and extension in its directory. As the client does not contain this file, it creates a new file, sets the timestamp value to -1.
3. Once the 'sync' button is pressed in Android console, the client sends the file name and timestamp of the file it needs to download from the server and starts waiting for the server response.
4. On the other side, once the server receives the file name and time stamp value from the client, the server checks the time stamp values of its own copy of file and the received timestamp.
5. As the server contains the recent file, the time stamp value of the server copy is greater and hence server starts writing the file to the client's output stream.
6. Once the entire file is received at the client side, the client initiates the connection termination by sending a "Bye" message to the server.
7. The server receives the "Bye" message and terminates the connection by closing the socket and the input, output stream objects created for file transfer.

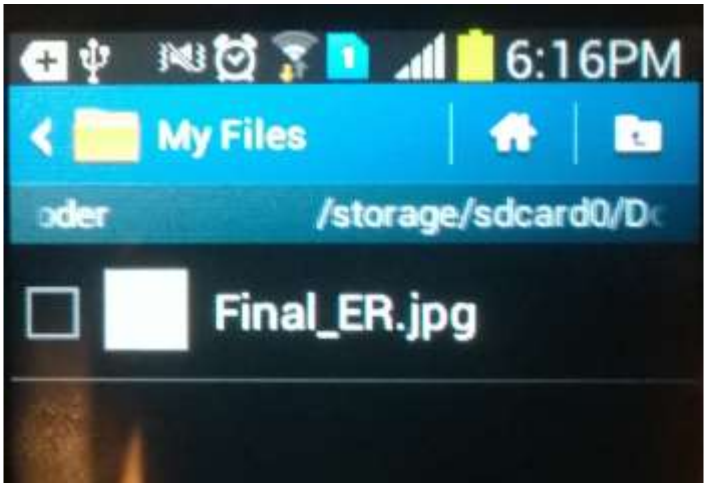
2.2.3 Scenario 3:

Client contains a file, wishes to upload the file to the server. This file does not exist in the server. The following steps are involved.

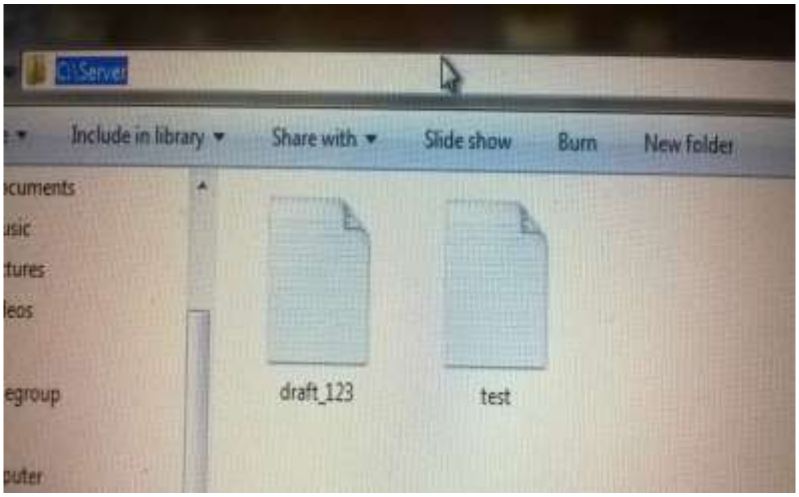
1. As the communication is through TCP, connection establishment occurs before the actual file transfer. Client (mobile device) initiates the connection establishment by entering the IP address and port number of the server (laptop) in the android console.
2. Once the connection has been established, the client checks whether there is a file with the same name and extension in its directory. As the client contains this file, it sets the timestamp value of the file to be the last modified date of the file.
3. Once the 'sync' button is pressed in Android console, the client sends the file name and timestamp of the file it needs to upload to the server and starts waiting for the server response.
4. On the other side, once the server receives the file name and time stamp value from the client, as the server does not contain the file; it creates a new file with the same name as the file name received from the client and sets the time stamp value of the file to -1.
5. As the client copy of the file is the recent, the time stamp value of the client copy is greater. The server sends the request for the copy of file. The client sends the entire copy of file to the server. Server receives the file in the client's input stream and writes it to its new copy of file created.
6. Once the entire file is sent from the client side, the client initiates the connection termination by sending a "Bye" message to the server.
7. The server receives the "Bye" message and terminates the connection by closing the socket and the input, output stream objects created for file transfer.

Demo of the scenario 3 to transfer a image file:

- The client contains an image file named Final_ER.jpg in the download folder.

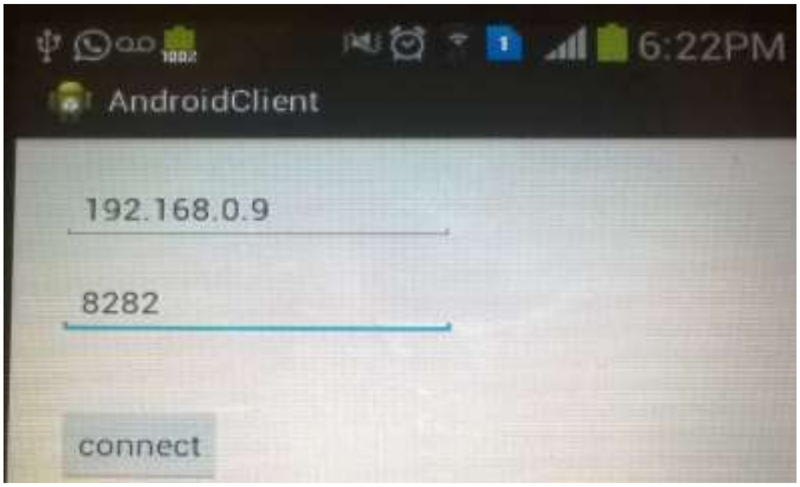


- The server does not contain this image file before the client-server communication.



C:\Server

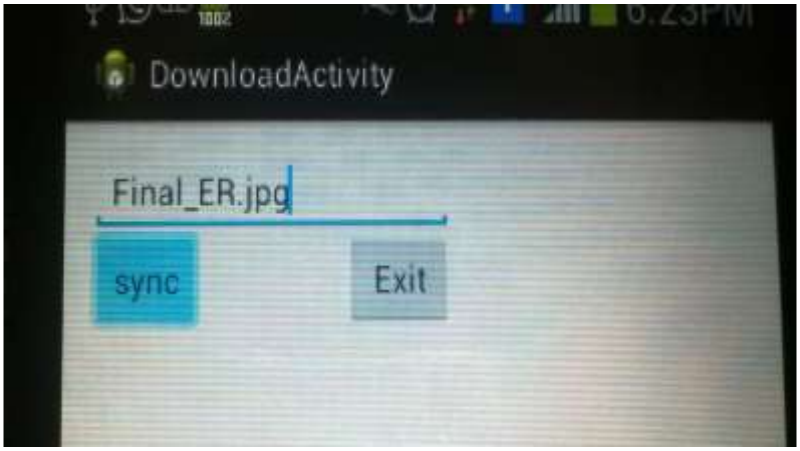
- The client initiates the connection request to the server by entering the IP address and of the laptop port number of the server program to the android console.



192.168.0.9

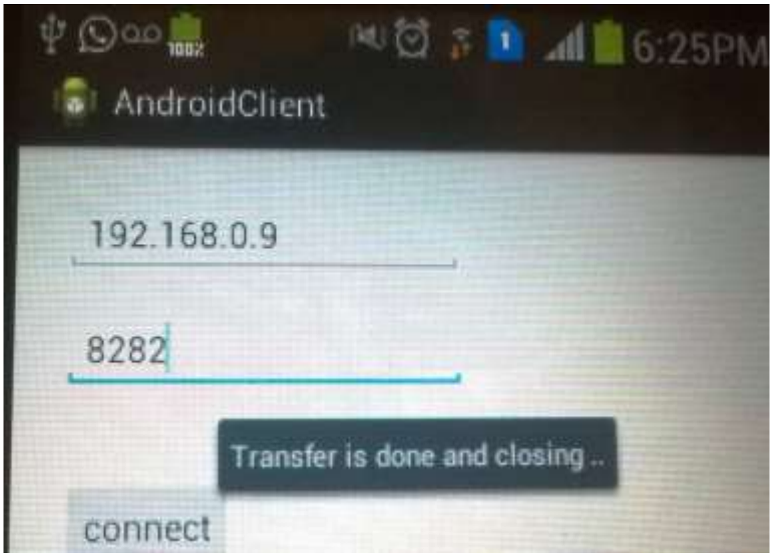
8282

- The client enters the image file name to be transferred in the file path and clicks on 'sync' button to initiate the file transfer process.



Final_ER.jpg

- ➢ Once the file is successfully uploaded to the server, file uploaded successfully message appears in the android console to intimate about the same to the client.

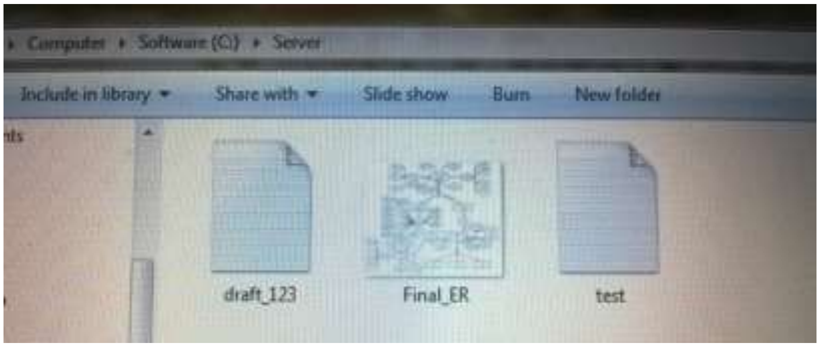


192.168.0.9

8282

Transfer is done and closing ..

- ➢ At this point of time, the server contains the File at the C:\ server directory.



2.2.4 Scenario 4:

Client contains a file, updates this file and wishes to upload the updated file to the server. This file already exists in the server. The following steps are involved.

1. As the communication is through TCP, connection establishment occurs before the actual file transfer. Client (mobile device) initiates the connection establishment by entering the IP address and port number of the server (laptop) in the android console.
2. Once the connection has been established, the client checks whether there is a file with the same name and extension in its directory. As the client contains this file, it sets the timestamp value of the file to be the last modified date of the file.
3. Once the 'sync' button is pressed in Android console, the client sends the file name and timestamp of the file it needs to upload to the server and starts waiting for the server response.
4. On the other side, once the server receives the file name and time stamp value from the client, the server checks the time stamp values of its own copy of file and the received timestamp.
5. As the client copy of the file is the recent, the time stamp value of the client copy is greater. The server sends the request for the copy of file. The client sends the entire copy of file to the server. Server receives the file in the client' input stream and writes it to its own copy of file.
6. Once the entire file is sent from the client side, the client initiates the connection termination by sending a "Bye" message to the server.
7. The server receives the "Bye" message and terminates the connection by closing the socket and the input, output stream objects created for file transfer.

2.2.5 Scenario 5:

Client at the mobile device attempts to sync a random file which neither exists in client nor at server. The following steps are involved.

1. As the communication is through TCP, connection establishment occurs before the actual file transfer. Client (mobile device) initiates the connection establishment by entering the IP address and port number of the server (laptop) in the android console.
2. Once the connection has been established, the client checks whether there is a file with the same name and extension in its directory. As the client does not contain this file, it create the file with the same name as specified in the file path and sets the timestamp value of the file to -1.
3. Once the 'sync' button is pressed in Android console, the client sends the random file name (which does not exist in client as well as server) and timestamp of the file and starts waiting for the server response.
4. On the other side, once the server receives the file name and time stamp value from the client, and the server checks whether it contains the copy of file. As it does not contain the copy of the file, it creates the file with the same name as sent by the client and writes the blank content to its copy of the file.

5. Once the entire file is sent from the client side, the client initiates the connection termination by sending a “Bye” message to the server.
6. The server receives the “Bye” message and terminates the connection by closing the socket and the input, output stream objects created for file transfer.
7. Before sending the Bye message and terminating the connection, if either the client or the server wishes to update the newly created file can do so within the same connection. The client/server who wishes to update the file first modifies the file in their file location and then now can use this connection created in step 1 to update the file to the other side.