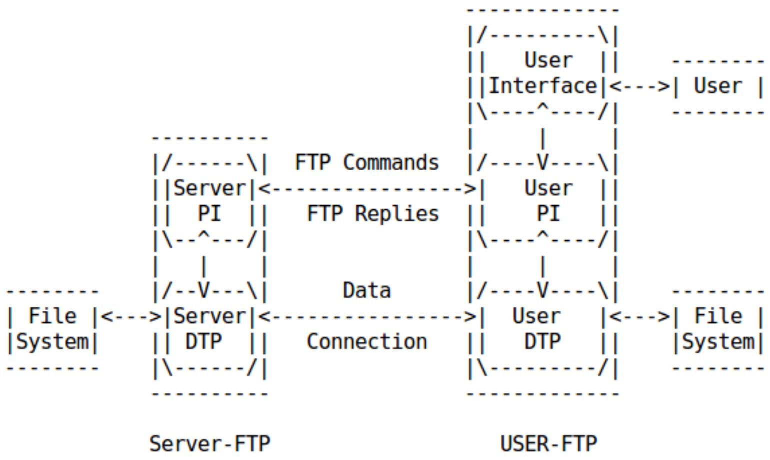
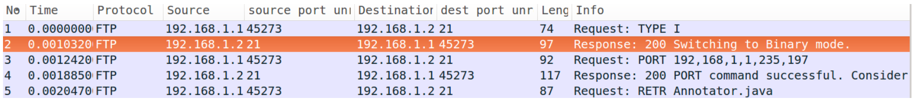
In this part, upon the two-ubuntu LAN that we built previously, we use FTP (File transfer protocol) to transfer a 7353 bytes file from host to server. And using Wireshark, I will give a detailed analysis on how the ftp files break into packets and into bits.



NOTES: 1. The data connection may be used in either direction.
2. The data connection need not exist all of the time.

Figure 1: 1

As can be seen from this picture, the data connection is established between the server DTP (data transfer process) and the user DTP, as for ftp commands and replies, they are transfered between server PI (protocol interpreter) and user PI.



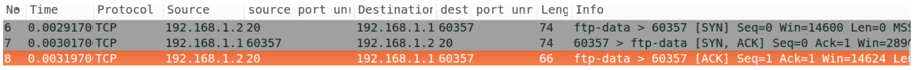
No	Time	Protocol	Source	source port uni	Destinatior	dest port unr	Leng	Info
1	0.0000000	FTP	192.168.1.1	45273	192.168.1.2	21	74	Request: TYPE I
2	0.0010320	FTP	192.168.1.2	21	192.168.1.1	45273	97	Response: 200 Switching to Binary mode.
3	0.0012420	FTP	192.168.1.1	45273	192.168.1.2	21	92	Request: PORT 192,168,1,1,235,197
4	0.0018850	FTP	192.168.1.2	21	192.168.1.1	45273	117	Response: 200 PORT command successful. Consider
5	0.0020470	FTP	192.168.1.1	45273	192.168.1.2	21	87	Request: RETR Annotator.java

Figure 2: 2

The above figure shows a typical ftp data connection establishment. 1, host request server for connection. 2, server responses code 200, meaning that connection permmitted. 3, and port 179,6 (45830) is to be used 4, use PASV, a mode where the client initiates the data connection. 5, ready to return the aimed file Annotator.java.

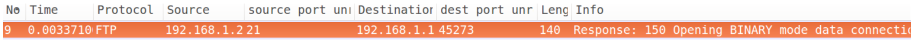
This image is a typical 3-way-hand-shaking TCP, to synthesize and establish a reliable connection.

This image shows that, code 150 means ”File status okay; about to open data connection.”



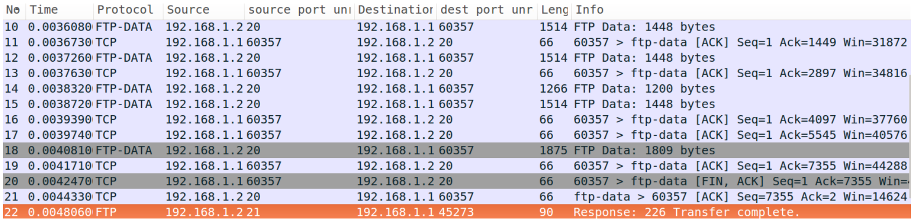
No	Time	Protocol	Source	source port uni	Destination	dest port unr	Leng	Info
6	0.0029170	TCP	192.168.1.2	20	192.168.1.1	60357	74	ftp-data > 60357 [SYN] Seq=0 Win=14600 Len=0 MS
7	0.0030170	TCP	192.168.1.1	60357	192.168.1.2	20	74	60357 > ftp-data [SYN, ACK] Seq=0 Ack=1 Win=289
8	0.0031970	TCP	192.168.1.2	20	192.168.1.1	60357	66	ftp-data > 60357 [ACK] Seq=1 Ack=1 Win=14624 Le

Figure 3: 3



No	Time	Protocol	Source	source port uni	Destination	dest port unr	Leng	Info
9	0.0033710	FTP	192.168.1.2	21	192.168.1.1	45273	140	Response: 150 Opening BINARY mode data connecti

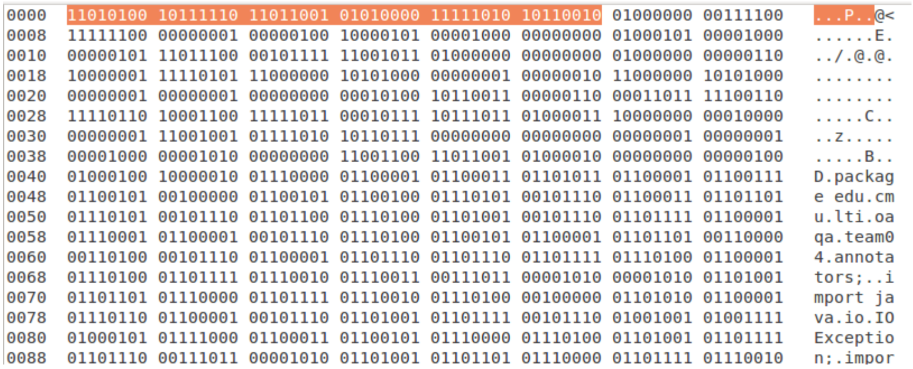
Figure 4: 4



No	Time	Protocol	Source	source port uni	Destination	dest port unr	Leng	Info
10	0.0036080	FTP-DATA	192.168.1.2	20	192.168.1.1	60357	1514	FTP Data: 1448 bytes
11	0.0036730	TCP	192.168.1.1	60357	192.168.1.2	20	66	60357 > ftp-data [ACK] Seq=1 Ack=1449 Win=31872
12	0.0037260	FTP-DATA	192.168.1.2	20	192.168.1.1	60357	1514	FTP Data: 1448 bytes
13	0.0037630	TCP	192.168.1.1	60357	192.168.1.2	20	66	60357 > ftp-data [ACK] Seq=1 Ack=2897 Win=34816
14	0.0038320	FTP-DATA	192.168.1.2	20	192.168.1.1	60357	1266	FTP Data: 1200 bytes
15	0.0038720	FTP-DATA	192.168.1.2	20	192.168.1.1	60357	1514	FTP Data: 1448 bytes
16	0.0039390	TCP	192.168.1.1	60357	192.168.1.2	20	66	60357 > ftp-data [ACK] Seq=1 Ack=4097 Win=37760
17	0.0039740	TCP	192.168.1.1	60357	192.168.1.2	20	66	60357 > ftp-data [ACK] Seq=1 Ack=5545 Win=40576
18	0.0040810	FTP-DATA	192.168.1.2	20	192.168.1.1	60357	1875	FTP Data: 1809 bytes
19	0.0041710	TCP	192.168.1.1	60357	192.168.1.2	20	66	60357 > ftp-data [ACK] Seq=1 Ack=7355 Win=44288
20	0.0042470	TCP	192.168.1.1	60357	192.168.1.2	20	66	60357 > ftp-data [FIN, ACK] Seq=1 Ack=7355 Win=
21	0.0044330	TCP	192.168.1.2	20	192.168.1.1	60357	66	ftp-data > 60357 [ACK] Seq=7355 Ack=2 Win=14624
22	0.0048060	FTP	192.168.1.2	21	192.168.1.1	45273	90	Response: 226 Transfer complete.

Figure 5: 5

Finally, we come to the file transfers, we can see that the files are transferred in several parts, and after each part of data transferred into user, user would send an ACK (acknowledgement) back to server, meaning that the user received the data successfully. At the end, code 226 means ”Closing data connection. Requested file action successful (for example, file transfer or file abort).”



```
0000  11010100 10111110 11011001 01010000 11111010 10110010 01000000 00111100   ...P..@<
0008  11111100 00000001 00000100 10000101 00001000 00000000 01000101 00001000   ......E.
0010  00000101 11011100 00101111 11001011 01000000 00000000 01000000 00000110   ../.@.@.
0018  10000001 11110101 11000000 10101000 00000001 00000010 11000000 10101000   ........
0020  00000001 00000001 00000000 00010100 10110011 00000110 00011011 11100110   ........
0028  11110110 10001100 11111011 00010111 10111011 01000011 10000000 00010000   .....C..
0030  00000001 11001001 01111010 10110111 00000000 00000000 00000001 00000001   ..z.....
0038  00001000 00001010 00000000 11001100 11011001 01000010 00000000 00000100   .....B..
0040  01000100 10000010 01110000 01100001 01100011 01101011 01100001 01100111   D.packag
0048  01100101 00100000 01100101 01100100 01110101 00101110 01100011 01101101   e edu.cm
0050  01110101 00101110 01101100 01110100 01101001 00101110 01101111 01100001   u.lti.oa
0058  01110001 01100001 00101110 01110100 01100101 01100001 01101101 00110000   qa.team0
0060  00110100 00101110 01100001 01101110 01101110 01101111 01110100 01100001   4.annota
0068  01110100 01101111 01110010 01110011 00111011 00001010 00001010 01101001   tors;..i
0070  01101101 01110000 01101111 01110010 01110100 00100000 01101010 01100001   mport ja
0078  01110110 01100001 00101110 01101001 01101111 00101110 01001001 01001111   va.io.IO
0080  01000101 01111000 01100011 01100101 01110000 01110100 01101001 01101111   Exceptio
0088  01101110 00111011 00001010 01101001 01101101 01110000 01101111 01110010   n;.impor
```

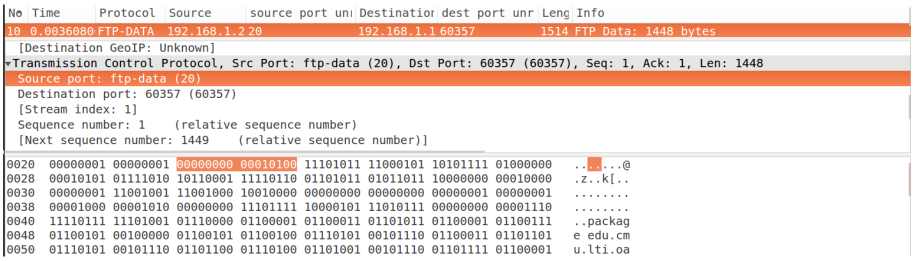
Figure 6: 6

Figure 6 shows one of the ftp-data packet, where bits of the packet can be seen.

First, how can we know if the packet is the ftp-data packet? We can check the source port (20 as ftp data) and destination port, and in the meantime check if there are bits after 66 bytes. (We will see the code inplementation of this soon)

Thus, similarly, for every ftp-data packet, we can grab those bits of data from the 66 bytes of the packet to the end.

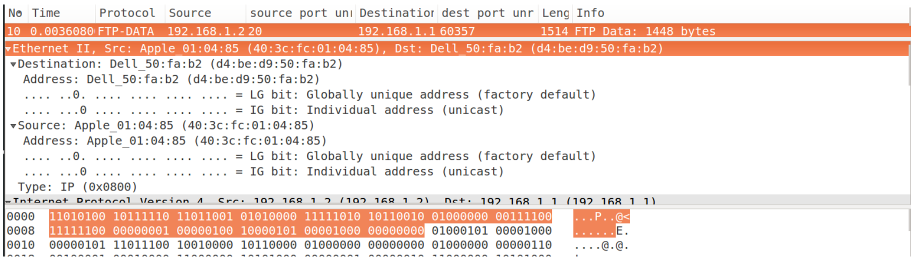
Now let's look into the bits of one of the ftp-data packet of the file (Annotator.java).



```
No Time       Protocol  Source       source port unı Destinatior dest port unr Leng Info
10 0.0036080 FTP-DATA  192.168.1.2 20               192.168.1.1 60357             1514 FTP Data: 1448 bytes
  [Destination GeoIP: Unknown]
▾Transmission Control Protocol, Src Port: ftp-data (20), Dst Port: 60357 (60357), Seq: 1, Ack: 1, Len: 1448
  Source port: ftp-data (20)
  Destination port: 60357 (60357)
  [Stream index: 1]
  Sequence number: 1    (relative sequence number)
  [Next sequence number: 1449    (relative sequence number)]

0020  00000001 00000001 00000000 00010100 11101011 11000101 10101111 01000000   .......@
0028  00010101 01111010 10110001 11110110 01101011 01011011 10000000 00010000   .z..k[..
0030  00000001 11001001 11001000 10010000 00000000 00000000 00000001 00000001   ........
0038  00001000 00001010 00000000 11101111 10000101 11010111 00000000 00001110   ........
0040  11110111 11101001 01110000 01100001 01100011 01101011 01100001 01100111   ..packag
0048  01100101 00100000 01100101 01100100 01110101 00101110 01100011 01101101   e edu.cm
0050  01110101 00101110 01101100 01110100 01101001 00101110 01101111 01100001   u.lti.oa
```

Figure 7: 7

For each packet:



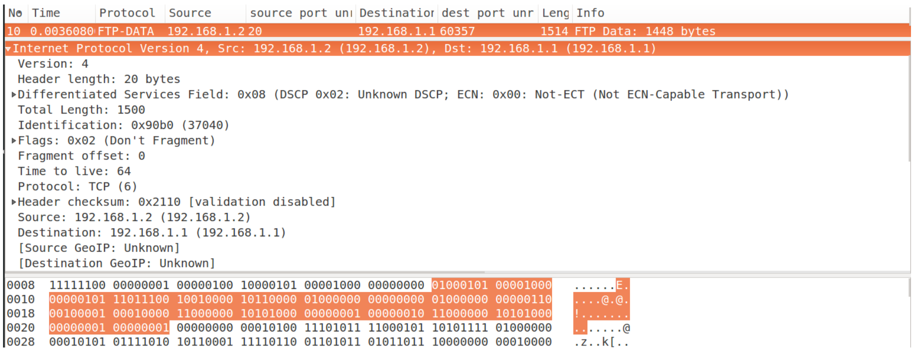
```
No Time       Protocol  Source       source port unı Destinatior dest port unr Leng Info
10 0.0036080 FTP-DATA  192.168.1.2 20               192.168.1.1 60357             1514 FTP Data: 1448 bytes
▾Ethernet II, Src: Apple_01:04:85 (40:3c:fc:01:04:85), Dst: Dell_50:fa:b2 (d4:be:d9:50:fa:b2)
 ▾Destination: Dell_50:fa:b2 (d4:be:d9:50:fa:b2)
   Address: Dell_50:fa:b2 (d4:be:d9:50:fa:b2)
   .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
   .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
 ▾Source: Apple_01:04:85 (40:3c:fc:01:04:85)
   Address: Apple_01:04:85 (40:3c:fc:01:04:85)
   .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
   .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  Type: IP (0x0800)
▸Internet Protocol Version 4, Src: 192.168.1.2 (192.168.1.2), Dst: 192.168.1.1 (192.168.1.1)

0000  11010100 10111110 11011001 01010000 11111010 10110010 01000000 00111100   ...P..@<
0008  11111100 00000001 00000100 10000101 00001000 00000000 01000101 00001000   ......E.
0010  00000101 11011100 10010000 10110000 01000000 00000000 01000000 00000110   ....@.@.
```

Figure 8: 8

1, first 6 bytes (0 5) are user mac address(d4:be:d9:50:fa:b2 here), 6 11 (6) bytes are server mac address (40:3c:fc:01:04:85), 12 13 (2) bytes are IP (0X0800).

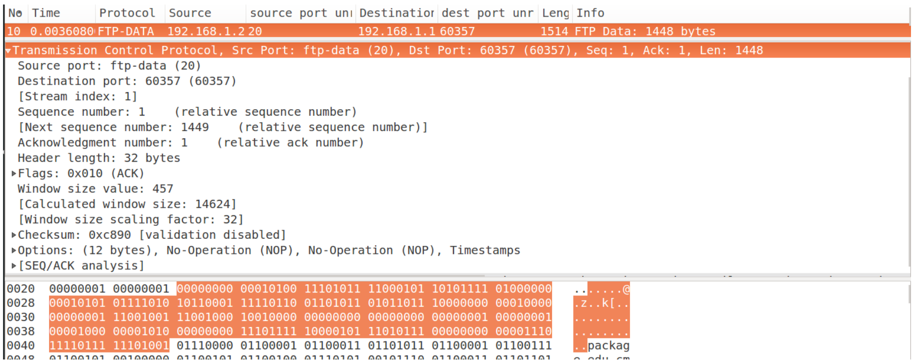
2, 14 byte is header length (20), ignore 15 byte, 16 17 (2) bytes are total length of data transferred (1500), 18 19 (2) bytes are identification (0x2fcb -¿ 12235), 20 21 (2) bytes are fragment offset (0), 22 is time to live (64), 23 byte is protocol used (6 -¿ TCP), 24 25 (2) bytes are header checksum (0x81f5 -¿ validation disabled), 26 29 (4) bytes are source GeoIP (unknown), 30 33 (4) bytes are destination GeoIP (unknown).

3, 34 35 (2) bytes are source port (20 -¿ ftp-data port), 36 37 (2) bytes are destination port (45830), 38 41 (4) bytes are sequence number (1), 42 45 (4) bytes are acknowledgment number (1), 46 47 (2) bytes are flags (0x010), 48 49 (2) bytes are window size value and scaling factor, 50 51 (2) bytes are checksum (0x7ab7 -¿ validation disabled), and the rest bytes all belong to TCP, until 62 65 (4) bytes are timestamp echo reply (279682).



```
No  Time       Protocol  Source       source port uni Destinatior dest port unr Leng Info
10  0.0036080 FTP-DATA  192.168.1.2 20              192.168.1.1 60357         1514 FTP Data: 1448 bytes
▾Internet Protocol Version 4, Src: 192.168.1.2 (192.168.1.2), Dst: 192.168.1.1 (192.168.1.1)
  Version: 4
  Header length: 20 bytes
 ▸Differentiated Services Field: 0x08 (DSCP 0x02: Unknown DSCP; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
  Total Length: 1500
  Identification: 0x90b0 (37040)
 ▸Flags: 0x02 (Don't Fragment)
  Fragment offset: 0
  Time to live: 64
  Protocol: TCP (6)
 ▸Header checksum: 0x2110 [validation disabled]
  Source: 192.168.1.2 (192.168.1.2)
  Destination: 192.168.1.1 (192.168.1.1)
  [Source GeoIP: Unknown]
  [Destination GeoIP: Unknown]
0008  11111100 00000001 00000100 10000101 00001000 00000000 01000101 00001000   ......E.
0010  00000101 11011100 10010000 10110000 01000000 00000000 01000000 00000110   ....@.@.
0018  00100001 00010000 11000000 10101000 00000001 00000010 11000000 10101000   !.......
0020  00000001 00000001 00000000 00010100 11101011 11000101 10101111 01000000   .......@
0028  00010101 01111010 10110001 11110110 01101011 01011011 10000000 00010000   .z..k[..
```

Figure 9: 9



```
No  Time       Protocol  Source       source port uni Destinatior dest port unr Leng Info
10  0.0036080 FTP-DATA  192.168.1.2 20              192.168.1.1 60357         1514 FTP Data: 1448 bytes
▾Transmission Control Protocol, Src Port: ftp-data (20), Dst Port: 60357 (60357), Seq: 1, Ack: 1, Len: 1448
  Source port: ftp-data (20)
  Destination port: 60357 (60357)
  [Stream index: 1]
  Sequence number: 1    (relative sequence number)
  [Next sequence number: 1449    (relative sequence number)]
  Acknowledgment number: 1    (relative ack number)
  Header length: 32 bytes
 ▸Flags: 0x010 (ACK)
  Window size value: 457
  [Calculated window size: 14624]
  [Window size scaling factor: 32]
 ▸Checksum: 0xc890 [validation disabled]
 ▸Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
 ▸[SEQ/ACK analysis]
0020  00000001 00000001 00000000 00010100 11101011 11000101 10101111 01000000   .......@
0028  00010101 01111010 10110001 11110110 01101011 01011011 10000000 00010000   .z..k[..
0030  00000001 11001001 11001000 10010000 00000000 00000000 00000001 00000001   ........
0038  00001000 00001010 00000000 11101111 10000101 11010111 00000000 00001110   ........
0040  11110111 11101001 01110000 01100001 01100011 01101011 01100001 01100111   ..packag
```

Figure 10: 10

4, then we see our data (1448 bytes), which takes a large part of the packet.

Thus, this analysis gives us the hint of how to abstract the bits of the file out of the packets flow.