

Progetto Wireshark:

Analisi di un Protocollo TCP-IP

Paolo De Mori

paolo.demori@studio.unibo.it

0001071000

INDICE:

1. [Introduzione](#)
2. [Analisi di un pacchetto TCP/IP](#)
 - a. [Analisi Header TCP](#)
 - b. [Analisi Header IP](#)
3. [Three Ways Handshake](#)
4. [Controllo di Flusso](#)
 - a. [Analisi del controllo di flusso nella cattura effettuata](#)
5. [Chiusura della connessione](#)
6. [Frammentazione](#)

Introduzione:

Per questo progetto ho deciso di analizzare una connessione ftp tramite Wireshark. Wireshark è un applicativo che permette di eseguire lo “*sniffing*”(ossia la cattura dei pacchetti) sulle interfacce di rete.

FTP (acronimo di File Transfer Protocol) è un protocollo applicativo che permette lo scambio di file.

FTP utilizza TCP come protocollo di trasporto e quindi è perfetto per l'analisi di pacchetti TCP-IP.

Dopo aver avviato Wireshark mi sono connesso a un server FTP tramite il comando **ftp** (fornito nativamente da windows), ho effettuato l'accesso come **anonymous**, e dopo aver scaricato un file(tramite il comando **get README**), ho chiuso la connessione tramite il comando **QUIT**.

A questo punto Wireshark ha catturato tutti i pacchetti utilizzati dalla mia interfaccia di rete per la connessione ftp. Posso quindi interrompere lo “*sniffing*”, tramite apposito pulsante.

Analisi di un Pacchetto TCP/IP:

Wireshark, permette un'approfondita analisi dei pacchetti raccolti durante la fase di “sniffing”.

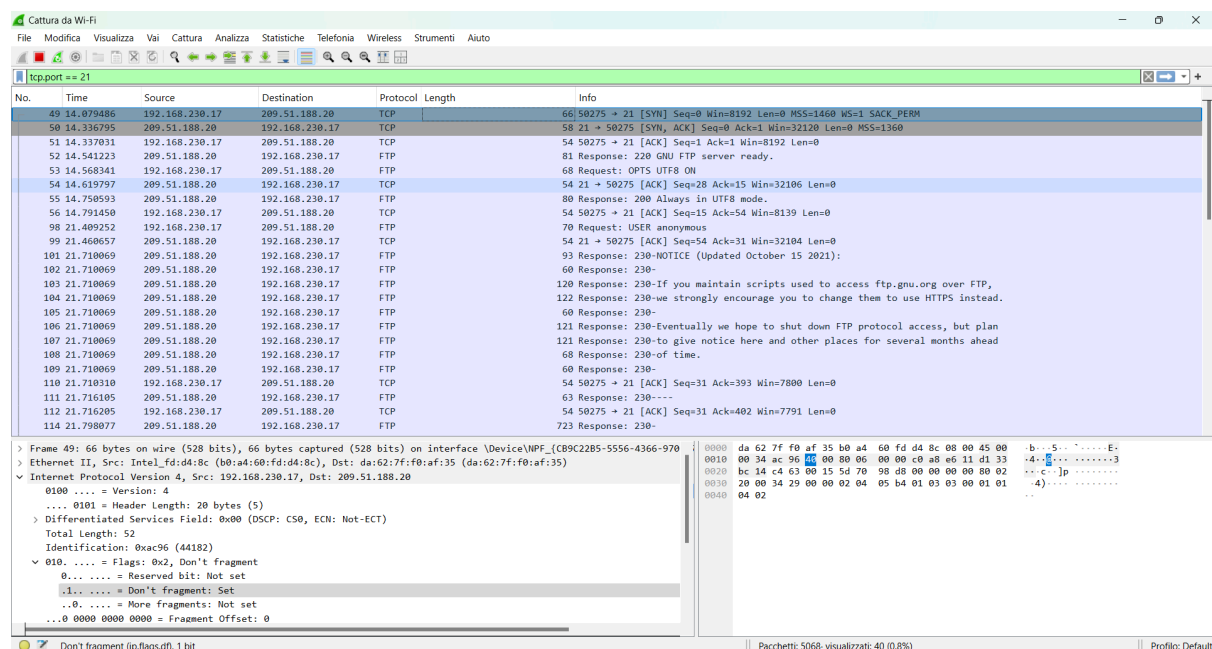
Il protocollo di livello applicativo “FTP” utilizza come protocollo di livello di trasporto “TCP” che a sua volta utilizza come protocollo di rete “ip”.

Lo standard di rete ISO/OSI stabilisce che, man mano che un pacchetto passa da un livello più alto a un livello più basso, venga aggiunto ad esso l'header del rispettivo protocollo di livello, in modo da permettere la corretta trasmissione.

Quindi utilizzando i pacchetti che il software ha intercettato durante la nostra connessione “FTP”, è possibile analizzare gli header “IP” e “TCP” di questi ultimi.

Applicando a Wireshark il filtro “**tcp.port == 21**”, è possibile mostrare solamente i pacchetti utilizzati dalla mia interfaccia di rete per la connessione “command” di protocolli “FTP” e procedere alla loro analisi.

Come pacchetto di esempio da analizzare ho selezionato il pacchetto **No.107** della cattura effettuata.



La cattura dei pacchetti filtrati con “tcp.port==21”

Analisi header TCP:

“TCP”, acronimo di “*Transmission Control Protocol*” è un protocollo del livello di trasporto definito come “*Connection Oriented*”, ossia richiede l'effettiva creazione di una connessione tra mittente e ricevente prima di iniziare il vero e proprio invio di dati.

“TCP” assicura il corretto scambio di dati, introducendo strategie di conferma della ricezione di dati e in caso di perdita di pacchetti la loro ritrasmissione.

Utilizzando Wireshark, è possibile scegliere un pacchetto (in questo caso il 107) e cliccando nella parte bassa dell'interfaccia sulla voce relativa al “*Transmission Control Protocol*” Wireshark permette di esaminare l’header “*TCP*” e vedere tutti i suoi campi:

50	14.336795	209.51.188.20	192.168.230.17	TCP	58 21 → 50275 [SYN, A
52	14.541223	209.51.188.20	192.168.230.17	FTP	81 Response: 220 GNU
54	14.619797	209.51.188.20	192.168.230.17	TCP	54 21 → 50275 [ACK] S
55	14.750593	209.51.188.20	192.168.230.17	FTP	80 Response: 200 Alwa
99	21.460657	209.51.188.20	192.168.230.17	TCP	54 21 → 50275 [ACK] S
101	21.710069	209.51.188.20	192.168.230.17	FTP	93 Response: 230-NOTI
102	21.710069	209.51.188.20	192.168.230.17	FTP	60 Response: 230-
103	21.710069	209.51.188.20	192.168.230.17	FTP	120 Response: 230-If y
104	21.710069	209.51.188.20	192.168.230.17	FTP	122 Response: 230-we s
105	21.710069	209.51.188.20	192.168.230.17	FTP	60 Response: 230-
106	21.710069	209.51.188.20	192.168.230.17	FTP	121 Response: 230-Even
107	21.710069	209.51.188.20	192.168.230.17	FTP	121 Response: 230-to g
108	21.710069	209.51.188.20	192.168.230.17	FTP	68 Response: 230-of t
109	21.710069	209.51.188.20	192.168.230.17	FTP	60 Response: 230-
111	21.716105	209.51.188.20	192.168.230.17	FTP	63 Response: 230----
114	21.798077	209.51.188.20	192.168.230.17	FTP	723 Response: 230-
158	38.443245	209.51.188.20	192.168.230.17	TCP	54 21 → 50275 [ACK] S
160	38.581553	209.51.188.20	192.168.230.17	FTP	105 Response: 200 PORT

> Frame 107: 121 bytes on wire (968 bits), 121 bytes captured (968 bits) on interface \Device\NPF_{CB9C22B5-5556-4366-
 > Ethernet II, Src: da:62:7f:f0:af:35 (da:62:7f:f0:af:35), Dst: Intel_fd:d4:8c (b0:a4:60:fd:d4:8c)
 > Internet Protocol Version 4, Src: 209.51.188.20, Dst: 192.168.230.17
 ✓ Transmission Control Protocol, Src Port: 21, Dst Port: 50275, Seq: 306, Ack: 31, Len: 67

- Source Port: 21
- Destination Port: 50275
- [Stream index: 5]
- > [Conversation completeness: Complete, WITH_DATA (31)]
- [TCP Segment Len: 67]
- Sequence Number: 306 (relative sequence number)
- Sequence Number (raw): 2665089788
- [Next Sequence Number: 373 (relative sequence number)]
- Acknowledgment Number: 31 (relative ack number)
- Acknowledgment number (raw): 1567660279
- 0101 = Header Length: 20 bytes (5)
- > Flags: 0x018 (PSH, ACK)
- Window: 32120

Come esaminare l'header tcp

Possiamo osservare che i 20 byte obbligatori dell'header TCP si dividono in:

1. 32 bit di Source Port e Destination Port:

```
Transmission Control Protocol, Src Port: 21, Dst Port: 50275, Seq: 306, Ack: 31, Len: 67
  Source Port: 21
  Destination Port: 50275
  [Stream index: 5]
  > [Conversation completeness: Complete, WITH_DATA (31)]
  [TCP Segment Len: 67]
  Sequence Number: 306 (relative sequence number)
  Sequence Number (raw): 2665089788
  [Next Sequence Number: 373 (relative sequence number)]
  Acknowledgment Number: 31 (relative ack number)
  Acknowledgment number (raw): 1567660279
  0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x018 (PSH, ACK)
  Window: 32120
  [Calculated window size: 32120]
  [Window size scaling factor: -2 (no window scaling used)]
  Checksum: 0xd2a7 [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
  > [Timestamps]
  > [SEQ/ACK analysis]
  TCP payload (67 bytes)
```

Source Port e Destination Port del pacchetto No.107

2. 32 bit di sequence number:

I dati inviati dai pacchetti tcp sono numerati utilizzando i byte. Il sequence number rappresenta il numero del primo byte dei dati che si stanno inviando. Il primo sequence number di una connessione chiamato “ISN” è stabilito al momento dell’instaurazione della connessione ed è in base a esso che poi verranno calcolati tutti gli altri.

Wireshark per semplificare la visualizzazione mostra

- l’effettivo numero di sequenza ossia “*ISN + il numero di byte inviati fino a quel momento+1(il primo byte che si sta inviando)*”
- la versione relativa ossia “*numero di sequenza del pacchetto - ISN*”
- il numero di sequenza relativo che ci si aspetta in caso di invio di un altro pacchetto di dati ossia “*(numero di sequenza + lunghezza segmento)-ISN*”.

Il numero di sequenza è utilizzato insieme al numero di acknowledgement per assicurare il corretto scambio di dati.

```

v Transmission Control Protocol, Src Port: 21, Dst Port: 50275, Seq: 306, Ack: 31, Len: 67
  Source Port: 21
  Destination Port: 50275
  [Stream index: 5]
  > [Conversation completeness: Complete, WITH_DATA (31)]
  [TCP Segment Len: 67]
  Sequence Number: 306 (relative sequence number)
  Sequence Number (raw): 2665089788
  [Next Sequence Number: 373 (relative sequence number)]
  Acknowledgment Number: 31 (relative ack number)
  Acknowledgment number (raw): 1567660279
  0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x018 (PSH, ACK)
  Window: 32120
  [Calculated window size: 32120]
  [Window size scaling factor: -2 (no window scaling used)]
  Checksum: 0xd2a7 [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
  > [Timestamps]
  > [SEQ/ACK analysis]
  TCP payload (67 bytes)

```

Sequence number del pacchetto No.107

3. 32 bit di acknowledgement number:

L'header tcp contiene un numero chiamato "Acknowledgement Number", che rappresenta quale dovrà essere il numero di sequenza del prossimo byte atteso dal mittente. Anche di quest'ultimo Wireshark fornisce una versione assoluta e una relativa. E' uno dei dati fondamentali per il controllo di flusso.

```

v Transmission Control Protocol, Src Port: 21, Dst Port: 50275, Seq: 306, Ack: 31, Len: 67
  Source Port: 21
  Destination Port: 50275
  [Stream index: 5]
  > [Conversation completeness: Complete, WITH_DATA (31)]
  [TCP Segment Len: 67]
  Sequence Number: 306 (relative sequence number)
  Sequence Number (raw): 2665089788
  [Next Sequence Number: 373 (relative sequence number)]
  Acknowledgment Number: 31 (relative ack number)
  Acknowledgment number (raw): 1567660279
  0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x018 (PSH, ACK)
  Window: 32120
  [Calculated window size: 32120]
  [Window size scaling factor: -2 (no window scaling used)]

```

Acknowledgement Number del pacchetto No.107

4. 4 bit di header length:

Rappresentano la lunghezza dell'header tcp, variabile in base alla lunghezza del campo "options".

```
Transmission Control Protocol, Src Port: 21, Dst Port: 50275, Seq: 306, Ack: 31, Len: 67
  Source Port: 21
  Destination Port: 50275
  [Stream index: 5]
  > [Conversation completeness: Complete, WITH_DATA (31)]
  [TCP Segment Len: 67]
  Sequence Number: 306 (relative sequence number)
  Sequence Number (raw): 2665089788
  [Next Sequence Number: 373 (relative sequence number)]
  Acknowledgment Number: 31 (relative ack number)
  Acknowledgment number (raw): 1567660279
  0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x018 (PSH, ACK)
  Window: 32120
  [Calculated window size: 32120]
  [Window size scaling factor: -2 (no window scaling used)]
```

Header length del pacchetto No.107

5. 6 bit reserved:

riservati per usi futuri

```
Transmission Control Protocol, Src Port: 21, Dst Port: 50275, Seq: 306, Ack: 31, Len: 67
  Source Port: 21
  Destination Port: 50275
  [Stream index: 5]
  > [Conversation completeness: Complete, WITH_DATA (31)]
  [TCP Segment Len: 67]
  Sequence Number: 306 (relative sequence number)
  Sequence Number (raw): 2665089788
  [Next Sequence Number: 373 (relative sequence number)]
  Acknowledgment Number: 31 (relative ack number)
  Acknowledgment number (raw): 1567660279
  0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x018 (PSH, ACK)
    000. .... = Reserved: Not set
    ...0 .... = Accurate ECN: Not set
    .... 0... = Congestion Window Reduced: Not set
    .... .0.. = ECN-Echo: Not set
    .... ..0. = Urgent: Not set
    .... ...1 = Acknowledgment: Set
    .... ....1... = Push: Set
    .... .... .0.. = Reset: Not set
    .... .... ..0. = Syn: Not set
    .... .... ...0 = Fin: Not set
    [TCP Flags: .....AP...]
  Window: 32120
```

Reserved bits del pacchetto No.107

6. 6 bit per flags:

Bit che possono essere posti a 1 per specificare un ruolo o specifica del pacchetto.

```
▼ Flags: 0x018 (PSH, ACK)
000. .... = Reserved: Not set
...0 .... = Accurate ECN: Not set
.... 0... = Congestion Window Reduced: Not set
.... .0.. = ECN-Echo: Not set
.... ..0. = Urgent: Not set
.... ...1 = Acknowledgment: Set
.... .... 1... = Push: Set
.... .... .0.. = Reset: Not set
.... .... ..0. = Syn: Not set
.... .... ...0 = Fin: Not set
[TCP Flags: .....AP...]
```

Flags del pacchetto No.107

7. 16 bit Window, 16bit di checksum e 16 bit di Urgent Pointer:

window definisce il massimo numero di byte che il ricevitore del pacchetto può inviare senza ricevere un ACK.

checksum è un meccanismo di controllo dell'errore.

Urgent Pointer contiene puntatori a elementi urgenti del pacchetto (utilizzato se flag URG è impostato ad 1)

```
Window: 32120
[Calculated window size: 32120]
[Window size scaling factor: -2 (no window scaling used)]
Checksum: 0xd2a7 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
```

Window Size del ricevitore del datagramma No.107

Mentre i bit di lunghezza variabile sono dati da Options (di lunghezza variabile) + padding per rendere l'header length un multiplo di 32.

```
.... ..0. .... = Urgent: Not set
.... ...1 .... = Acknowledgment: Set
.... .... 1... = Push: Set
.... .... .0.. = Reset: Not set
.... .... ..0. = Syn: Not set
.... .... ...0 = Fin: Not set
[TCP Flags: .....AP...]
Window: 32120
[Calculated window size: 32120]
[Window size scaling factor: -2 (no window scaling used)]
Checksum: 0xd2a7 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
> [Timestamps]
> [SEQ/ACK analysis]
TCP payload (67 bytes)
```

Option del datagramma No.107

Resoconto del pacchetto in analisi:

Analizzando il pacchetto selezionato, possiamo affermare che:

Il pacchetto no.107 E' un pacchetto che al livello di trasporto:

- Ha come protocollo TCP
- Ha come porta sorgente la porta 21

- Ha come porta di destinazione la porta 50275
- Utilizzando il sequence number possiamo affermare che, il primo byte di dati che invia è il numero 165 dall'inizio della connessione
- Ha il payload TCP di 67 byte, infatti Wireshark ci segnala che l'eventuale prossimo sequence number è il 233 ($165+67+1$)
- Ha i flag Push e Acknowledge attivi (Quindi sta effettuando piggybacking)
- La window size del mittente è di 32120, che rappresenta il massimo numero di byte che il ricevitore può ricevere prima di un pacchetto di acknowledge

Analisi header IP:

Il protocollo IP è un protocollo di rete, e ha come obiettivo quello di permettere l'instradamento dei pacchetti attraverso reti anche di diversa tipologia.

È un protocollo di tipo "*connectionless*", fornisce possibilità di frammentazione dei suoi pacchetti chiamati "*datagrammi*", ma non presenta alcun tipo di controllo di flusso o di sequenza che invece sono delegati agli strati superiori dello standard ISO/OSI.

Per identificare univocamente gli interlocutori attraverso la rete, ognuno è dotato di un indirizzo, formato convenzionalmente da 32 bit, rappresentati come 4 numeri decimali da 0 a 255.

Similmente a quanto fatto per l'analisi dell'header tcp, cliccando sul pacchetto desiderato (in questo caso il no. 107) e selezionando nella parte bassa dello schermo la voce "*Internet Protocol*" posso vedere le informazioni sull'header IP del pacchetto.

tcp.port == 21						
No.	Time	Source	Destination	Protocol	Length	Info
50	14.336795	209.51.188.20	192.168.230.17	TCP		58 21 → 50275 [SYN, ACK]
52	14.541223	209.51.188.20	192.168.230.17	FTP		81 Response: 220 GNU FTP
54	14.619797	209.51.188.20	192.168.230.17	TCP		54 21 → 50275 [ACK] Seq=
55	14.750593	209.51.188.20	192.168.230.17	FTP		80 Response: 200 Always
99	21.460657	209.51.188.20	192.168.230.17	TCP		54 21 → 50275 [ACK] Seq=
101	21.710069	209.51.188.20	192.168.230.17	FTP		93 Response: 230-NOTICE
102	21.710069	209.51.188.20	192.168.230.17	FTP		60 Response: 230-
103	21.710069	209.51.188.20	192.168.230.17	FTP		120 Response: 230-If you
104	21.710069	209.51.188.20	192.168.230.17	FTP		122 Response: 230-we strc
105	21.710069	209.51.188.20	192.168.230.17	FTP		60 Response: 230-
106	21.710069	209.51.188.20	192.168.230.17	FTP		121 Response: 230-Eventua
107	21.710069	209.51.188.20	192.168.230.17	FTP		121 Response: 230-to give
108	21.710069	209.51.188.20	192.168.230.17	FTP		68 Response: 230-of time


```

> Frame 107: 121 bytes on wire (968 bits), 121 bytes captured (968 bits) on interface \Device\NPF_{CB9C22B5-5556-4366-
> Ethernet II, Src: da:62:7f:f0:af:35 (da:62:7f:f0:af:35), Dst: Intel_fd:d4:8c (b0:a4:60:fd:d4:8c)
> Internet Protocol Version 4, Src: 209.51.188.20, Dst: 192.168.230.17
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x20 (DSCP: CS1, ECN: Not-ECT)
    Total Length: 107
    Identification: 0xe24e (57934)
  > 000. .... = Flags: 0x0
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..0. .... = More fragments: Not set
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 51
    Protocol: TCP (6)
    Header Checksum: 0x711c [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 209.51.188.20
    Destination Address: 192.168.230.17
    [Stream index: 6]

```

IP Header del datagramma No.107

La lunghezza minima di un header IP è di 20 byte.

I byte obbligatori sono suddivisi in:

1. 4 bit di versione e 4 bit di lunghezza dell'header in parole da 32 bit:

Wireshark mostra la "header length" espressa in byte con affianco tra parentesi il numero di parole da 32 bit

```

> Internet Protocol Version 4, Src: 209.51.188.20, Dst: 192.168.230.17
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x20 (DSCP: CS1, ECN: Not-ECT)
    Total Length: 107
    Identification: 0xe24e (57934)
  > 000. .... = Flags: 0x0
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..0. .... = More fragments: Not set
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 51
    Protocol: TCP (6)
    Header Checksum: 0x711c [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 209.51.188.20
    Destination Address: 192.168.230.17
    [Stream index: 6]

```

Versione e Header Length del datagramma No.107

2. 8 bit per specificare il tipo di servizio richiesto e 16 bit per la lunghezza totale del datagramma:

```
Internet Protocol Version 4, Src: 209.51.188.20, Dst: 192.168.230.17
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x20 (DSCP: CS1, ECN: Not-ECT)
  Total Length: 107
  Identification: 0xe24e (57934)
  000. .... = Flags: 0x0
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..0. .... = More fragments: Not set
    ...0 0000 0000 0000 = Fragment Offset: 0
  Time to Live: 51
  Protocol: TCP (6)
  Header Checksum: 0x711c [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 209.51.188.20
  Destination Address: 192.168.230.17
  [Stream index: 6]
```

Tipo di Servizio e Total Length del datagramma No.107

3. 32 bit utilizzati soprattutto per la frammentazione, suddivisi in:

- 16 bit di “Identification”: un numero che rappresenta univocamente il datagramma e permette di identificare a che datagramma corrisponde un certo offset.
- 3 bit per le flags rispettivamente:
 - 1 bit sempre a 0
 - “Don’t fragment flag”(DF), 0 il datagramma si può frammentare, 1 non si può frammentare
 - “More fragments flag”(MF), 0 è l’ultimo frammento, 1 ci sono altri frammenti del datagramma
- 13 bit fragment offset, rappresenta la posizione del frammento nel datagramma espressa come distanza dall’inizio(0)

```
Internet Protocol Version 4, Src: 209.51.188.20, Dst: 192.168.230.17
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x20 (DSCP: CS1, ECN: Not-ECT)
  Total Length: 107
  Identification: 0xe24e (57934)
  000. .... = Flags: 0x0
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..0. .... = More fragments: Not set
    ...0 0000 0000 0000 = Fragment Offset: 0
  Time to Live: 51
  Protocol: TCP (6)
  Header Checksum: 0x711c [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 209.51.188.20
```

Flags del datagramma No.107

4.

- 8 Bit per il “time to live”(TTL), ossia il massimo numero di nodi attraversabili dal pacchetto. Tale valore viene decrementato da ogni nodo che attraversa, una volta a 0 il pacchetto viene distrutto
- 8 Bit per rappresentare il protocollo di livello superiore che utilizza IP

- c. **16 Bit per il calcolo del checksum**(meccanismo per verificare l'integrità dei dati) dell'header, che viene ricalcolato ad ogni passaggio per un nodo

```
Internet Protocol Version 4, Src: 209.51.188.20, Dst: 192.168.230.17
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x20 (DSCP: CS1, ECN: Not-ECT)
    Total Length: 79
    Identification: 0xe23f (57919)
  000. .... = Flags: 0x0
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..0. .... = More fragments: Not set
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 51
    Protocol: TCP (6)
    Header Checksum: 0x7147 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 209.51.188.20
    Destination Address: 192.168.230.17
```

Time To Live, Protocollo di trasporto e Checksum del datagramma No.107

5. **32 Bit per l'indirizzo di partenza del pacchetto e 32 Bit per l'indirizzo di arrivo:**

```
Internet Protocol Version 4, Src: 209.51.188.20, Dst: 192.168.230.17
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x20 (DSCP: CS1, ECN: Not-ECT)
    Total Length: 79
    Identification: 0xe23f (57919)
  000. .... = Flags: 0x0
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..0. .... = More fragments: Not set
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 51
    Protocol: TCP (6)
    Header Checksum: 0x7147 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 209.51.188.20
    Destination Address: 192.168.230.17
    [Stream index: 6]
```

Source Address e Destination Address del datagramma No.107

Resoconto del pacchetto in analisi:

Analizzando il pacchetto selezionato, possiamo affermare che:

Il pacchetto no.107 E' un pacchetto che allo strato di Rete:

- Utilizza il protocollo IP versione 4
- Ha come indirizzo di partenza 209.51.188
- Ha come indirizzo di destinazione 192.168.230.17
- Ha un header lungo 20 bytes con una lunghezza totale del datagramma di 107 bytes
- Non è parte di un datagramma frammentato, in quanto ha come flag Mf 0 e come offset 0.
- Può attraversare 51 nodi prima di arrivare a destinazione senza essere distrutto
- E' sfruttato per il trasporto di un pacchetto TCP.

Three Ways Handshake:

La three ways handshake è il meccanismo utilizzato per instaurare connessioni TCP, tramite l'invio di tre pacchetti che utilizzando i flag *ACK* e *SYN*, il “*sequence number*”(il numero di sequenza del primo byte dei dati che si stanno inviando)e “*acknowledgement number*”.

Tramite Wireshark è facile identificare i 3 “pacchetti della three ways handshake”, poiché il software evidenzia i flag presenti(inserendoli nelle parentesi quadre),il “*sequence number*”(tramite la sigla “seq”) e “*acknowledgement number*”(con la sigla Ack).

49	14.079486	192.168.230.17	209.51.188.20	TCP	66	50275 → 21 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=1 SACK_PERM
50	14.336795	209.51.188.20	192.168.230.17	TCP	58	21 → 50275 [SYN, ACK] Seq=0 Ack=1 Win=32120 Len=0 MSS=1360
51	14.337031	192.168.230.17	209.51.188.20	TCP	54	50275 → 21 [ACK] Seq=1 Ack=1 Win=8192 Len=0

i Tre pacchetti della “three ways handshake”

Il primo pacchetto della “three ways handshake” è caratterizzato dalla presenza del solo flag SYN impostato a 1.

Analizzando il pacchetto TCP notiamo che in questo header i flags sono tutti a 0 tranne il bit che rappresenta il flag SYN impostato a 1. Questa configurazione, rappresenta proprio la richiesta di instaurazione di una connessione TCP.

Analisi dei flag del primo pacchetto:

```
Flags: 0x002 (SYN)
000... = Reserved: Not set
...0... = Accurate ECN: Not set
...0... = Congestion Window Reduced: Not set
...0... = ECH-Echo: Not set
...0... = Urgent: Not set
...0... = Acknowledgment: Not set
...0... = Push: Not set
...0... = Reset: Not set
...1... = SYN: Set
...0... = Fin: Not set
```

analisi dei flag del primo pacchetto

Il protocollo TCP non numera i segmenti, ma i singoli byte attraverso i numeri di sequenza.

Il primo numero di sequenza della connessione è fondamentale in quanto definisce il numero da cui si inizierà il conteggio dei byte trasferiti tramite il protocollo.

In una connessione TCP il primo numero di sequenza definito come *ISN* è specificato all'interno del primo pacchetto di ogni connessione.

49	14.079486	192.168.230.17	209.51.188.20	TCP	66	50275 → 21 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=1 SACK_PERM
50	14.336795	209.51.188.20	192.168.230.17	TCP	58	21 → 50275 [SYN, ACK] Seq=0 Ack=1 Win=32120 Len=0 MSS=1360

> Frame 49: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF{CB9C22B5-5556-4366-970C-}	0000	da 62 7f f0 af 35 b0 a4 60 fd d4
> Ethernet II, Src: Intel_fd:d4:8c (b0:a4:60:fd:d4:8c), Dst: da:62:7f:f0:af:35 (da:62:7f:f0:af:35)	0010	00 34 ac 96 40 00 80 06 00 00 c0
> Internet Protocol Version 4, Src: 192.168.230.17, Dst: 209.51.188.20	0020	bc 14 c4 63 00 15 5d 70 98 d8 00
> Transmission Control Protocol, Src Port: 50275, Dst Port: 21, Seq: 0, Len: 0	0030	20 00 34 29 00 00 02 04 05 b4 01
	0040	04 02

Source Port: 50275
Destination Port: 21
[Stream index: 5]
[Conversation completeness: Complete, WITH_DATA (31)]
[TCP Segment Len: 0]
Sequence Number: 0 (relative sequence number)
Sequence Number (raw): 1567660248
[Next Sequence Number: 1 (relative sequence number)]
Acknowledgment Number: 0
Acknowledgment number (raw): 0
1000 = Header Length: 32 bytes (8)
Flags: 0x002 (SYN)
Window: 8192
[Calculated window size: 8192]
Checksum: 0x3429 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Operation (NOP),
[Timestamps]

ISN

Il secondo pacchetto che compone il “three ways handshake” viene inviato dal ricevente del primo pacchetto. Oltre quindi a presentare “*source port*” e “*destination port*” invertiti rispetto al primo; tale pacchetto è caratterizzato da flag SYN impostato a 1, flag ACK impostato ad 1, *ISN* del rispettivo canale e “*acknowledge number*” pari al numero di sequenza del primo pacchetto +1. Sempre verificabile nell’header TCP nella voce Transmission Control Protocol.

No.	Time	Source	Destination	Protocol	Length	Info
49	14.079486	192.168.230.17	209.51.188.20	TCP	66	50275 → 21 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=1 SACK_PERM
50	14.336795	209.51.188.20	192.168.230.17	TCP	58	21 → 50275 [SYN, ACK] Seq=0 Ack=1 Win=32120 Len=0 MSS=1360
51	14.337031	192.168.230.17	209.51.188.20	TCP	54	50275 → 21 [ACK] Seq=1 Ack=1 Win=8192 Len=0
52	14.541223	209.51.188.20	192.168.230.17	FTP	81	Response: 220 GNU FTP server ready.
53	14.568341	192.168.230.17	209.51.188.20	FTP	68	Request: OPTS UTF8 ON
54	14.619797	209.51.188.20	192.168.230.17	TCP	54	21 → 50275 [ACK] Seq=28 Ack=15 Win=32106 Len=0
55	14.750593	209.51.188.20	192.168.230.17	FTP	80	Response: 200 Always in UTF8 mode.
56	14.791450	192.168.230.17	209.51.188.20	TCP	54	50275 → 21 [ACK] Seq=15 Ack=54 Win=8139 Len=0
98	21.409252	192.168.230.17	209.51.188.20	FTP	70	Request: USER anonymous
99	21.460657	209.51.188.20	192.168.230.17	TCP	54	21 → 50275 [ACK] Seq=54 Ack=31 Win=32104 Len=0
101	21.710069	209.51.188.20	192.168.230.17	FTP	93	Response: 230-NOTICE (Updated October 15 2021):
102	21.710069	209.51.188.20	192.168.230.17	FTP	60	Response: 230-
103	21.710069	209.51.188.20	192.168.230.17	FTP	120	Response: 230-If you maintain scripts used to access ftp.gnu.org over FTP,
104	21.710069	209.51.188.20	192.168.230.17	FTP	122	Response: 230-we strongly encourage you to change them to use HTTPS instead.
105	21.710069	209.51.188.20	192.168.230.17	FTP	60	Response: 230-
106	21.710069	209.51.188.20	192.168.230.17	FTP	121	Response: 230-Eventually we hope to shut down FTP protocol access, but plan
107	21.710069	209.51.188.20	192.168.230.17	FTP	121	Response: 230-to give notice here and other places for several months ahead
108	21.710069	209.51.188.20	192.168.230.17	FTP	68	Response: 230-of time.
109	21.710069	209.51.188.20	192.168.230.17	FTP	60	Response: 230-
110	21.710310	192.168.230.17	209.51.188.20	TCP	54	50275 → 21 [ACK] Seq=31 Ack=393 Win=7800 Len=0
111	21.716105	209.51.188.20	192.168.230.17	FTP	63	Response: 230----
112	21.716205	192.168.230.17	209.51.188.20	TCP	54	50275 → 21 [ACK] Seq=31 Ack=402 Win=7791 Len=0
114	21.798077	209.51.188.20	192.168.230.17	FTP	723	Response: 230-

Transmission Control Protocol, Src Port: 21, Dst Port: 50275, Seq: 0, Ack: 1, Len: 0
Source Port: 21
Destination Port: 50275
[Stream index: 5]
[Conversation completeness: Complete, WITH_DATA (31)]
[TCP Segment Len: 0]
Sequence Number: 0 (relative sequence number)
Sequence Number (raw): 2665089482
[Next Sequence Number: 1 (relative sequence number)]
Acknowledgment Number: 1 (relative ack number)
Acknowledgment number (raw): 1567660249
0110 = Header Length: 24 bytes (6)
Flags: 0x012 (SYN, ACK)

0000 b0 a4 60 fd d4 8c da 62 7f f0 af 35 00 00 45 20b...5..E
0010 00 2c e2 09 00 00 33 06 71 a0 d1 33 bc 14 c0 a8 ,...3. q-3...
0020 e6 11 00 15 c4 63 9e da 09 ca 5d 70 98 d8 60 12c...p...
0030 7d 78 83 98 00 00 02 04 05 50 }x.....P

Acknowledge Number del secondo pacchetto pari al sequence number +1 del primo

```

0110 .... = Header Length: 24 bytes (6)
✓ Flags: 0x012 (SYN, ACK)
  000. .... = Reserved: Not set
  ...0 .... = Accurate ECN: Not set
  .... 0... = Congestion Window Reduced: Not set
  .... .0.. = ECN-Echo: Not set
  .... ..0. = Urgent: Not set
  .... ...1 = Acknowledgment: Set
  .... .... = Push: Not set
  .... ....0.. = Reset: Not set
  > .... ....1. = Syn: Set
  .... ....0 = Fin: Not set
[TCP Flags: .....A..S.]

```

Analisi dei flag del secondo pacchetto

Il terzo pacchetto rappresenta la conferma della ricezione dell'avvenuta connessione e segnala al ricevente l'imminente invio di dati. Infatti il terzo pacchetto ha il bit SYN settato a 0, il bit ACK impostato a 1, il numero di sequenza pari al numero di sequenza del primo pacchetto che verrà inviato e "l' *acknowledgement number*" pari al numero di sequenza inviato dal mittente +1.

No.	Time	Source	Destination	Protocol	Length	Info
49	14.079486	192.168.230.17	209.51.188.20	TCP	66	50275 → 21 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=1 SACK_PERM
50	14.336795	209.51.188.20	192.168.230.17	TCP	58	21 → 50275 [SYN, ACK] Seq=0 Ack=1 Win=32120 Len=0 MSS=1360
51	14.337031	192.168.230.17	209.51.188.20	TCP	54	50275 → 21 [ACK] Seq=1 Ack=1 Win=8192 Len=0
52	14.541223	209.51.188.20	192.168.230.17	FTP	81	Response: 230 GNU FTP server ready.
53	14.568341	192.168.230.17	209.51.188.20	FTP	68	Request: OPTS UTF8 ON
54	14.619797	209.51.188.20	192.168.230.17	TCP	54	21 → 50275 [ACK] Seq=28 Ack=15 Win=32106 Len=0
55	14.750593	209.51.188.20	192.168.230.17	FTP	80	Response: 200 Always in UTF8 mode.
56	14.791450	192.168.230.17	209.51.188.20	TCP	54	50275 → 21 [ACK] Seq=15 Ack=54 Win=8139 Len=0
98	21.409252	192.168.230.17	209.51.188.20	FTP	70	Request: USER anonymous
99	21.460657	209.51.188.20	192.168.230.17	TCP	54	21 → 50275 [ACK] Seq=54 Ack=31 Win=32104 Len=0
101	21.710069	209.51.188.20	192.168.230.17	FTP	93	Response: 230-NOTICE (Updated October 15 2021):
102	21.710069	209.51.188.20	192.168.230.17	FTP	60	Response: 230-
103	21.710069	209.51.188.20	192.168.230.17	FTP	120	Response: 230-If you maintain scripts used to access ftp.gnu.org over FTP,
104	21.710069	209.51.188.20	192.168.230.17	FTP	122	Response: 230-we strongly encourage you to change them to use HTTPS instead.
105	21.710069	209.51.188.20	192.168.230.17	FTP	60	Response: 230-
106	21.710069	209.51.188.20	192.168.230.17	FTP	121	Response: 230-Eventually we hope to shut down FTP protocol access, but plan
107	21.710069	209.51.188.20	192.168.230.17	FTP	121	Response: 230-to give notice here and other places for several months ahead
108	21.710069	209.51.188.20	192.168.230.17	FTP	68	Response: 230-of time.
109	21.710069	209.51.188.20	192.168.230.17	FTP	60	Response: 230-
110	21.710310	192.168.230.17	209.51.188.20	TCP	54	50275 → 21 [ACK] Seq=31 Ack=393 Win=7800 Len=0
111	21.716105	209.51.188.20	192.168.230.17	FTP	63	Response: 230----
112	21.716205	192.168.230.17	209.51.188.20	TCP	54	50275 → 21 [ACK] Seq=31 Ack=402 Win=7791 Len=0
114	21.798077	209.51.188.20	192.168.230.17	FTP	723	Response: 230-


```

0101 .... = Header Length: 20 bytes (5)
✓ Flags: 0x010 (ACK)
  000. .... = Reserved: Not set
  ...0 .... = Accurate ECN: Not set
  .... 0... = Congestion Window Reduced: Not set
  .... .0.. = ECN-Echo: Not set
  .... ..0. = Urgent: Not set
  .... ...1 = Acknowledgment: Set
  .... .... = Push: Not set
  .... ....0.. = Reset: Not set
  .... ....0. = Syn: Not set
  .... ....0 = Fin: Not set
[TCP Flags: .....A....]

```

0000 da 62 7f f0 af 35 b0 a4 60 fd d4 8c 08 00 45 00 b...5...E
0010 00 28 ac 97 40 00 80 06 00 00 c0 a8 e6 11 d1 33 (...@...3
0020 bc 14 c4 63 00 15 5d 70 98 d9 9e da 09 cb 50 10 ...c...p.....
0030 20 00 34 1d 00 004...

Acknowledgement number del terzo pacchetto uguale al sequence number del primo più 1

49	14.079486	192.168.230.17	209.51.188.20	TCP	66	50275 → 21 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=1 SACK_PERM
50	14.336795	209.51.188.20	192.168.230.17	TCP	58	21 → 50275 [SYN, ACK] Seq=0 Ack=1 Win=32120 Len=0 MSS=1360
51	14.337031	192.168.230.17	209.51.188.20	TCP	54	50275 → 21 [ACK] Seq=0 Ack=1 Win=8192 Len=0
52	14.541223	209.51.188.20	192.168.230.17	FTP	81	Response: 220 GNU FTP server ready.
53	14.568341	192.168.230.17	209.51.188.20	FTP	68	Request: OPTS UTF8 ON
54	14.619797	209.51.188.20	192.168.230.17	TCP	54	21 → 50275 [ACK] Seq=28 Ack=15 Win=32106 Len=0
55	14.750593	209.51.188.20	192.168.230.17	FTP	80	Response: 200 Always in UTF8 mode.
56	14.791450	192.168.230.17	209.51.188.20	TCP	54	50275 → 21 [ACK] Seq=15 Ack=54 Win=8139 Len=0
98	21.409252	192.168.230.17	209.51.188.20	FTP	70	Request: USER anonymous
99	21.460657	209.51.188.20	192.168.230.17	TCP	54	21 → 50275 [ACK] Seq=54 Ack=31 Win=32104 Len=0
101	21.710069	209.51.188.20	192.168.230.17	FTP	93	Response: 230-NOTICE (Updated October 15 2021):
102	21.710069	209.51.188.20	192.168.230.17	FTP	60	Response: 230-
103	21.710069	209.51.188.20	192.168.230.17	FTP	120	Response: 230-If you maintain scripts used to access ftp.gnu.or
104	21.710069	209.51.188.20	192.168.230.17	FTP	122	Response: 230-we strongly encourage you to change them to use t
105	21.710069	209.51.188.20	192.168.230.17	FTP	60	Response: 230-
106	21.710069	209.51.188.20	192.168.230.17	FTP	121	Response: 230-Eventually we hope to shut down FTP protocol acce
107	21.710069	209.51.188.20	192.168.230.17	FTP	121	Response: 230-to give notice here and other places for several
108	21.710069	209.51.188.20	192.168.230.17	FTP	68	Response: 230-of time.
109	21.710069	209.51.188.20	192.168.230.17	FTP	60	Response: 230-
110	21.710310	192.168.230.17	209.51.188.20	TCP	54	50275 → 21 [ACK] Seq=31 Ack=393 Win=7800 Len=0
111	21.716105	209.51.188.20	192.168.230.17	FTP	63	Response: 230----
112	21.716205	192.168.230.17	209.51.188.20	TCP	54	50275 → 21 [ACK] Seq=31 Ack=402 Win=7791 Len=0
114	21.798077	209.51.188.20	192.168.230.17	FTP	723	Response: 230-

Source Port: 21
Destination Port: 50275
[Stream index: 5]
> [Conversation completeness: Complete, WITH_DATA (31)]
[TCP Segment Len: 27]
Sequence Number: 1 (relative sequence number)
Sequence Number (raw): 2665089483
[Next Sequence Number: 28 (relative sequence number)]
Acknowledgment Number: 1 (relative ack number)
Acknowledgment number (raw): 1567660249
0101 = Header Length: 20 bytes (5)
Flags: 0x018 (PSH, ACK)
0000 = Reserved: Not set

0000 b0 a4 60 fd d4 8c da 62 7f f0 af
0010 00 43 e2 0c 00 00 33 06 71 86 d1
0020 e6 11 00 15 c4 63 9e da 09 cb 5d
0030 7d 78 97 f0 00 00 32 32 30 20 17
0040 50 20 73 65 72 76 65 72 20 72 65
0050 0a

Sequence Number (tcp.seq): 4 bvt

Pacchetti: 22450- visualizzati: 50 (0.2%) - s

Sequence number del terzo pacchetto è uguale al primo pacchetto contenente dati

Nel caso in cui durante una connessione ftp, si presenta la necessità di inviare uno o più file c'è la necessità di instaurare un' altra connessione TCP su una porta TCP diversa dalla 21.

Infatti modificando il filtro della nostra cattura Wireshark, visualizzando tutti i pacchetti in cui o come sorgente o come destinazione dei pacchetti ip è presente il server FTP, potrà facilmente notare un'altra three ways handshake utilizzata per creare la connessione "data".

tcp.port == 21 || ip.dst == 209.51.188.20 || ip.src == 209.51.188.20

Il filtro per visualizzare anche la connessione "data"

No.	Time	Source	Destination	Protocol	Length	Info
110	21.710310	192.168.230.17	209.51.188.20	TCP	54	50275 → 21 [ACK] Seq=31 Ack=393 Win=7800 Len=0
111	21.716105	209.51.188.20	192.168.230.17	FTP	63	Response: 230----
112	21.716205	192.168.230.17	209.51.188.20	TCP	54	50275 → 21 [ACK] Seq=31 Ack=402 Win=7791 Len=0
114	21.716977	209.51.188.20	192.168.230.17	FTP	723	Response: 230----
115	21.844136	192.168.230.17	209.51.188.20	TCP	54	50275 → 21 [ACK] Seq=31 Ack=1071 Win=7122 Len=0
157	38.387617	192.168.230.17	209.51.188.20	TCP	83	Request: PORT 192,168,230,17,196,100
158	38.443245	209.51.188.20	192.168.230.17	TCP	54	21 → 50275 [ACK] Seq=1071 Ack=60 Win=32992 Len=0
160	38.581553	209.51.188.20	192.168.230.17	FTP	105	Response: 200 PORT command successful. Consider using PASV.
161	38.601098	192.168.230.17	209.51.188.20	FTP	67	Request: RETR README
162	38.641695	209.51.188.20	192.168.230.17	TCP	54	21 → 50275 [ACK] Seq=1122 Ack=73 Win=32107 Len=0
163	38.810906	209.51.188.20	192.168.230.17	TCP	74	20 → 50276 [SYN] Seq=0 Win=64240 Len=0 MSS=1360 SACK_PERM TSval=1870834440 TSecr=0 WS=128
164	38.811278	192.168.230.17	209.51.188.20	TCP	74	50276 → 20 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM TSval=1936388 TSecr=1870834440
166	38.996342	209.51.188.20	192.168.230.17	TCP	66	20 → 50276 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1870834646 TSecr=1936388
167	38.996342	209.51.188.20	192.168.230.17	FTP-DL	1414	FTP Data: 1348 bytes (PORT 192,168,230,17,196,100)
168	38.996994	209.51.188.20	192.168.230.17	FTP-DL	1414	FTP Data: 1348 bytes (PORT 192,168,230,17,196,100)
169	38.999994	209.51.188.20	192.168.230.17	FTP-DL	118	FTP Data: 52 bytes (PORT 192,168,230,17,196,100)
170	38.999994	209.51.188.20	192.168.230.17	TCP	66	20 → 50276 [FIN, ACK] Seq=2749 Ack=1 Win=64256 Len=0 TSval=1870834647 TSecr=1936388
171	38.999994	209.51.188.20	192.168.230.17	FTP	120	Response: 150 Opening BINARY mode data connection for README (2748 bytes).
172	38.999247	192.168.230.17	209.51.188.20	TCP	66	50276 → 20 [ACK] Seq=1 Ack=2750 Win=1048576 Len=0 TSval=1936469 TSecr=1870834646
173	39.049427	192.168.230.17	209.51.188.20	TCP	54	50275 → 21 [ACK] Seq=73 Ack=1188 Win=7005 Len=0
174	39.067238	192.168.230.17	209.51.188.20	TCP	66	50276 → 20 [FIN, ACK] Seq=1 Ack=2750 Win=1048576 Len=0 TSval=1936564 TSecr=1870834646
175	39.220632	209.51.188.20	192.168.230.17	FTP	78	Response: 226 Transfer complete.
176	39.220632	209.51.188.20	192.168.230.17	TCP	66	20 → 50276 [ACK] Seq=2750 Ack=2 Win=64256 Len=0 TSval=1870834888 TSecr=1936564
177	39.271243	192.168.230.17	209.51.188.20	TCP	54	50275 → 21 [ACK] Seq=73 Ack=1212 Win=6981 Len=0
194	50.209315	192.168.230.17	209.51.188.20	FTP	60	Request: QUIT
197	44.604061	209.51.188.20	192.168.230.17	TCP	54	21 → 50275 [ACK] Seq=1212 Ack=79 Win=32114 Len=0
202	44.716322	209.51.188.20	192.168.230.17	FTP	68	Response: 221 Goodbye.
203	44.716322	209.51.188.20	192.168.230.17	TCP	54	21 → 50275 [FIN, ACK] Seq=1226 Ack=79 Win=32120 Len=0
204	44.716507	192.168.230.17	209.51.188.20	TCP	54	50275 → 21 [ACK] Seq=79 Ack=1227 Win=6967 Len=0
205	44.730364	192.168.230.17	209.51.188.20	TCP	54	50275 → 21 [FIN, ACK] Seq=79 Ack=1227 Win=6967 Len=0
206	44.753827	209.51.188.20	192.168.230.17	TCP	54	21 → 50275 [ACK] Seq=1227 Ack=80 Win=32120 Len=0

La three ways handshake della connessione “data” in dettaglio

Controllo di Flusso:

Il protocollo TCP, presenta un meccanismo di controllo di flusso che garantisce l'integrità dei dati e la consistenza della connessione nel tempo.

Il controllo di flusso del protocollo TCP, si basa su un meccanismo chiamato “a finestra scorrevole” che utilizza i cosiddetti pacchetti di “*acknowledgement*”, caratterizzati dal flag *ACK* impostato ad 1.

Tali pacchetti sono inviati dai ricevitori di pacchetti e confermano la corretta ricezione dei dati.

I pacchetti di “*acknowledgement*” possono essere inviati singolarmente oppure contenuti all’interno di pacchetti che trasmettono dati in modo da ottimizzare il numero di pacchetti inviati, questa tecnica è chiamata “*piggybacking*”.

L'invio di pacchetti di *acknowledgement* inoltre può essere ritardato per permettere l'invio di meno pacchetti.

I campi aggiornati da un ricevente che riceve un pacchetto di “*acknowledgement*” sono:

- Acknowledge Number
- Window Size

Window Size:

Il meccanismo a finestra scorrevole si basa sul principio che, un mittente smetta di inviare dati, se non riceve un pacchetto di “*acknowledgement*” per un certo numero di byte o di segmenti.

Il numero di byte o di segmenti che un mittente è in grado di inviare, senza dover ricevere una conferma di ricezione è identificabile nel campo dell'header TCP

denominato “*Window Size*” ed è aggiornato ad ogni invio di pacchetto di conferma di ricezione.

Acknowledgement Number:

Ad ogni ricezione di un pacchetto con flag *ACK* impostato ad uno il mittente dei dati, riceve la conferma dei dati ricevuti. Questa conferma è possibile grazie all'utilizzo del campo dei pacchetti tcp denominato “*Acknowledgement Number*”.

“*Acknowledgement Number*” rappresenta il numero di sequenza del primo byte che il mittente del pacchetto si attende di ricevere nel caso di ricezione di nuovi dati, precisamente rappresentato dall’ *ISN* ossia il primo numero di sequenza impostato all’avvio della connessione + il numero di byte ricevuti fino a quel momento+1.

Il controllo di flusso inoltre è permesso da un timer chiamato “*persist timer*”, che inizia il suo conteggio al momento dell’invio di un pacchetto e se scade senza che il mittente abbia ricevuto una conferma di ricezione del pacchetto, segnala al mittente la necessità di ri-invio dei dati.

Nel caso in cui un pacchetto arrivi al ricevitore in un ordine “errato”, rispetto alla sequenza di dati, il ricevitore applica una strategia sempre basata sui pacchetti di “*acknowledgement*”.

Alla ricezione di un pacchetto fuori ordine:

1. Se già ricevuto, lo scarta
2. Se mancano uno o più segmenti, se il buffer lo permette, il ricevente memorizza il segmento e ri-invia “*Acknowledgement*” dell’ultimo segmento in ordine. In caso riceva i pacchetti mancanti, invierà le conferme necessarie, se invece scade il “*persist timer*”, senza ricevere *ack* , il mittente ri-inverrà il pacchetto.

Analisi del controllo di flusso nella cattura effettuata:

Nella nostra cattura è possibile identificare facilmente il meccanismo di controllo di flusso TCP.

Prendiamo per esempio in analisi i pacchetti No. 53,54, 55 e 56

53	14.568341	192.168.230.17	209.51.188.20	FTP	68 Request: OPTS UTF8 ON
54	14.619797	209.51.188.20	192.168.230.17	TCP	54 21 → 50275 [ACK] Seq=28 Ack=15 Win=32106 Len=0
55	14.750593	209.51.188.20	192.168.230.17	FTP	80 Response: 200 Always in UTF8 mode.
56	14.791450	192.168.230.17	209.51.188.20	TCP	54 50275 → 21 [ACK] Seq=15 Ack=54 Win=8139 Len=0

Possiamo verificare che al momento dell’invio del pacchetto 53, il mittente “192.168.230.17” ha come window size “8165”.

53	14.568341	192.168.230.17	209.51.188.20	FTP	68	Request: OPTS UTF8 C
54	14.619797	209.51.188.20	192.168.230.17	TCP	54	21 → 50275 [ACK] Seq
55	14.750593	209.51.188.20	192.168.230.17	FTP	80	Response: 200 Always
56	14.791450	192.168.230.17	209.51.188.20	TCP	54	50275 → 21 [ACK] Seq
98	21.409252	192.168.230.17	209.51.188.20	FTP	70	Request: USER anonym
99	21.460657	209.51.188.20	192.168.230.17	TCP	54	21 → 50275 [ACK] Seq
101	21.710069	209.51.188.20	192.168.230.17	FTP	93	Response: 230-NOTICE
102	21.710069	209.51.188.20	192.168.230.17	FTP	60	Response: 230-
103	21.710069	209.51.188.20	192.168.230.17	FTP	120	Response: 230-Tf vou

> Internet Protocol Version 4, Src: 192.168.230.17, Dst: 209.51.188.20
 ✓ Transmission Control Protocol, Src Port: 50275, Dst Port: 21, Seq: 1, Ack: 28, Len: 14
 Source Port: 50275
 Destination Port: 21
 [Stream index: 5]
 > [Conversation completeness: Complete, WITH_DATA (31)]
 [TCP Segment Len: 14]
 Sequence Number: 1 (relative sequence number)
 Sequence Number (raw): 1567660249
 [Next Sequence Number: 15 (relative sequence number)]
 Acknowledgment Number: 28 (relative ack number)
 Acknowledgment number (raw): 2665089510
 0101 = Header Length: 20 bytes (5)
 > Flags: 0x018 (PSH, ACK)
 Window: 8165
 [Calculated window size: 8165]
 [Window size scaling factor: -2 (no window scaling used)]
 Checksum: 0x342b [unverified]
 [Checksum Status: Unverified]
 Urgent Pointer: 0
 > [Timestamps]
 > [SEQ/ACK analysis]

Successivamente possiamo notare come l'interlocutore "209.51.188.20" invia due pacchetti.

Il pacchetto 54 è un pacchetto di *acknowledgement* puro, quindi con 0 byte di dati trasportati(ovviamente escluso header).

53	14.568341	192.168.230.17	209.51.188.20	FTP	68	Request: OPTS UTF8 C
54	14.619797	209.51.188.20	192.168.230.17	TCP	54	21 → 50275 [ACK] Seq
55	14.750593	209.51.188.20	192.168.230.17	FTP	80	Response: 200 Always
56	14.791450	192.168.230.17	209.51.188.20	TCP	54	50275 → 21 [ACK] Seq
98	21.409252	192.168.230.17	209.51.188.20	FTP	70	Request: USER anonym
99	21.460657	209.51.188.20	192.168.230.17	TCP	54	21 → 50275 [ACK] Seq
101	21.710069	209.51.188.20	192.168.230.17	FTP	93	Response: 230-NOTICE
102	21.710069	209.51.188.20	192.168.230.17	FTP	60	Response: 230-
103	21.710069	209.51.188.20	192.168.230.17	FTP	120	Response: 230-Tf vou

> Internet Protocol Version 4, Src: 209.51.188.20, Dst: 192.168.230.17
 ✓ Transmission Control Protocol, Src Port: 21, Dst Port: 50275, Seq: 28, Ack: 15, Len: 0
 Source Port: 21
 Destination Port: 50275
 [Stream index: 5]
 > [Conversation completeness: Complete, WITH_DATA (31)]
 [TCP Segment Len: 0]
 Sequence Number: 28 (relative sequence number)
 Sequence Number (raw): 2665089510
 [Next Sequence Number: 28 (relative sequence number)]
 Acknowledgment Number: 15 (relative ack number)
 Acknowledgment number (raw): 1567660263
 0101 = Header Length: 20 bytes (5)
 ✓ Flags: 0x010 (ACK)
 000. = Reserved: Not set
 ...0 = Accurate ECN: Not set
 ... 0... = Congestion Window Reduced: Not set
 0.. = ECN-Echo: Not set
 0. = Urgent: Not set
 1 = Acknowledgment: Set

Il pacchetto 55 invece, è un pacchetto che presenta 26 byte di dati inviati.

55	14.750593	209.51.188.20	192.168.230.17	FTP	80	Response: 200 Always
56	14.791450	192.168.230.17	209.51.188.20	TCP	54	50275 → 21 [ACK] Seq
98	21.409252	192.168.230.17	209.51.188.20	FTP	70	Request: USER anonym
99	21.460657	209.51.188.20	192.168.230.17	TCP	54	21 → 50275 [ACK] Seq
101	21.710069	209.51.188.20	192.168.230.17	FTP	93	Response: 230-NOTIC
102	21.710069	209.51.188.20	192.168.230.17	FTP	60	Response: 230-
103	21.710069	209.51.188.20	192.168.230.17	FTP	120	Response: 230-If v

> Frame 55: 80 bytes on wire (640 bits), 80 bytes captured (640 bits) on interface \Device\NPF_{CB9C22B5-5556-4366-970}

> Ethernet II, Src: da:62:7f:f0:af:35 (da:62:7f:f0:af:35), Dst: Intel_fd:d4:8c (b0:a4:60:fd:d4:8c)

> Internet Protocol Version 4, Src: 209.51.188.20, Dst: 192.168.230.17

✓ Transmission Control Protocol, Src Port: 21, Dst Port: 50275, Seq: 28, Ack: 15, Len: 26

Source Port: 21

Destination Port: 50275

[Stream index: 5]

> [Conversation completeness: Complete, WITH_DATA (31)]

[TCP Segment Len: 26]

Sequence Number: 28 (relative sequence number)

Sequence Number (raw): 2665089510

[Next Sequence Number: 54 (relative sequence number)]

Acknowledgment Number: 15 (relative ack number)

Acknowledgment number (raw): 1567660263

0101 = Header Length: 20 bytes (5)

✓ Flags: 0x018 (PSH, ACK)

000. = Reserved: Not set

...0 = Accurate ECN: Not set

.... 0... = Congestion Window Reduced: Not set

.... .0.. = ECN-Echo: Not set

.... ..0. = Urgent: Not set

.... ...1 = Acknowledgment: Set

Quindi possiamo aspettarci che la grandezza del buffer del ricevitore prima dell'invio di un ack sia $8165 - (26 + 0)$, quindi 8139.

Effettivamente controllando il pacchetto numero 56, verifichiamo la correttezza della nostra ipotesi.

56	14.791450	192.168.230.17	209.51.188.20	TCP	54	50275 → 21 [ACK] S
98	21.409252	192.168.230.17	209.51.188.20	FTP	70	Request: USER anon
99	21.460657	209.51.188.20	192.168.230.17	TCP	54	21 → 50275 [ACK] S
101	21.710069	209.51.188.20	192.168.230.17	FTP	93	Response: 230-NOTI
102	21.710069	209.51.188.20	192.168.230.17	FTP	60	Response: 230-
103	21.710069	209.51.188.20	192.168.230.17	FTP	120	Response: 230-Tf v


```

[Stream index: 5]
> [Conversation completeness: Complete, WITH_DATA (31)]
[TCP Segment Len: 0]
Sequence Number: 15 (relative sequence number)
Sequence Number (raw): 1567660263
[Next Sequence Number: 15 (relative sequence number)]
Acknowledgment Number: 54 (relative ack number)
Acknowledgment number (raw): 2665089536
0101 .... = Header Length: 20 bytes (5)
✓ Flags: 0x010 (ACK)
  000. .... = Reserved: Not set
  ...0 .... = Accurate ECN: Not set
  ....0... = Congestion Window Reduced: Not set
  ....0... = ECN-Echo: Not set
  ....0... = Urgent: Not set
  ....1... = Acknowledgment: Set
  ....0... = Push: Not set
  ....0... = Reset: Not set
  ....0... = Syn: Not set
  ....0... = Fin: Not set
[TCP Flags: .....A....]
Window: 8139

```

Chiusura della Connessione:

Analizzando la connessione possiamo notare il meccanismo utilizzato dalle connessioni TCP in caso di richiesta da parte di uno dei due interlocutori di chiusura di una connessione.

Il protocollo TCP tenta sempre di realizzare delle “soft release”, ossia tenta sempre di chiudere le connessioni in modo ordinato, senza perdere dati.

Le richieste di chiusura di una connessione avvengono tramite pacchetti il cui flag FIN è impostato ad 1.

Nel nostro caso possiamo, utilizzando il filtro “tcp.port == 21” e la notazione di Wireshark, che nella sezione “info” evidenzia i flag attivi, identificare i pacchetti il cui flag FIN è impostato ad 1.

No.	Time	Source	Destination	Protocol	Length	Info
101	21.710069	209.51.188.20	192.168.230.17	FTP		93 Response: 230-NOTICE (Updated October 15 2021):
102	21.710069	209.51.188.20	192.168.230.17	FTP		60 Response: 230-
103	21.710069	209.51.188.20	192.168.230.17	FTP		120 Response: 230-If you maintain scripts used to access ftp.gnu.org over FTP,
104	21.710069	209.51.188.20	192.168.230.17	FTP		122 Response: 230-we strongly encourage you to change them to use HTTPS instead.
105	21.710069	209.51.188.20	192.168.230.17	FTP		60 Response: 230-
106	21.710069	209.51.188.20	192.168.230.17	FTP		121 Response: 230-Eventually we hope to shut down FTP protocol access, but plan
107	21.710069	209.51.188.20	192.168.230.17	FTP		121 Response: 230-to give notice here and other places for several months ahead
108	21.710069	209.51.188.20	192.168.230.17	FTP		60 Response: 230-of time.
109	21.710069	209.51.188.20	192.168.230.17	FTP		60 Response: 230-
110	21.710310	192.168.230.17	209.51.188.20	TCP		54 50275 → 21 [ACK] Seq=31 Ack=393 Win=7800 Len=0
111	21.716105	209.51.188.20	192.168.230.17	FTP		63 Response: 230----
112	21.716205	192.168.230.17	209.51.188.20	TCP		54 50275 → 21 [ACK] Seq=31 Ack=402 Win=7791 Len=0
114	21.798077	209.51.188.20	192.168.230.17	FTP		723 Response: 230-
115	21.844136	192.168.230.17	209.51.188.20	TCP		54 50275 → 21 [ACK] Seq=31 Ack=1071 Win=7122 Len=0
157	38.387617	192.168.230.17	209.51.188.20	FTP		83 Request: PORT 192,168,230,17,196,100
158	38.443245	209.51.188.20	192.168.230.17	TCP		54 21 → 50275 [ACK] Seq=1071 Ack=60 Win=32092 Len=0
160	38.581553	209.51.188.20	192.168.230.17	FTP		105 Response: 200 PORT command successful. Consider using PASV.
161	38.601098	192.168.230.17	209.51.188.20	FTP		67 Request: RETR README
162	38.641695	209.51.188.20	192.168.230.17	TCP		54 21 → 50275 [ACK] Seq=1122 Ack=73 Win=32107 Len=0
171	38.999094	209.51.188.20	192.168.230.17	FTP		120 Response: 150 Opening BINARY mode data connection for README (2748 bytes).
173	39.049427	192.168.230.17	209.51.188.20	TCP		54 50275 → 21 [ACK] Seq=73 Ack=1188 Win=7005 Len=0
175	39.220632	209.51.188.20	192.168.230.17	FTP		78 Response: 226 Transfer complete.
177	39.271243	192.168.230.17	209.51.188.20	TCP		54 50275 → 21 [ACK] Seq=73 Ack=1212 Win=6981 Len=0
194	44.510935	192.168.230.17	209.51.188.20	FTP		60 Request: QUIT
197	44.604061	209.51.188.20	192.168.230.17	TCP		54 21 → 50275 [ACK] Seq=1212 Ack=79 Win=32114 Len=0
202	44.716322	209.51.188.20	192.168.230.17	FTP		60 Response: 221 Goodbye.
203	44.716322	209.51.188.20	192.168.230.17	TCP		54 21 → 50275 [FIN, ACK] Seq=1226 Ack=79 Win=32120 Len=0
204	44.716507	192.168.230.17	209.51.188.20	TCP		54 50275 → 21 [ACK] Seq=79 Ack=1227 Win=6967 Len=0
205	44.730364	192.168.230.17	209.51.188.20	TCP		54 50275 → 21 [FIN, ACK] Seq=79 Ack=1227 Win=6967 Len=0
206	44.753827	209.51.188.20	192.168.230.17	TCP		54 21 → 50275 [ACK] Seq=1227 Ack=80 Win=32120 Len=0

Pacchetti utilizzati per chiudere la connessione TCP

In alternativa possiamo aggiungere al filtro “tcp.port==21” il filtro “tcp.flags.fin == 1”, legati dall’ and logico “&&” per mostrare solo i pacchetti della connessione ftp che richiedono la chiusura della connessione.

tcp.port == 21 && tcp.flags.fin == 1						
No.	Time	Source	Destination	Protocol	Length	Info
203	44.716322	209.51.188.20	192.168.230.17	TCP		54 21 → 50275 [FIN, ACK] Seq=1226 Ack=79 Win=32120 Len=0
205	44.730364	192.168.230.17	209.51.188.20	TCP		54 50275 → 21 [FIN, ACK] Seq=79 Ack=1227 Win=6967 Len=0

Solo i pacchetti con flag FIN impostato ad 1

Il nostro è un caso di “chiusura normale” della connessione tcp, in cui l’interlocutore all’indirizzo 209.51.188.20 invia un pacchetto con flag FIN = 1, notificando l’intenzione di chiudere la connessione. Il ricevente, ricevuto il pacchetto, invia un pacchetto di “*acknowledgement*” (con flag ACK = 1) e successivamente procede a inviare a sua volta un pacchetto con flag *FIN*=1. La connessione termina definitivamente con l’invio di un pacchetto di “*acknowledgement*” da parte dell’interlocutore che ha per primo manifestato la volontà di chiudere la connessione.

Frammentazione:

La Frammentazione è una tecnologia nativamente inclusa nel protocollo IP. Permette la trasmissione di datagrammi più grandi della *MTU* (*Maximum Transmission Unit* ossia la massima grandezza di un datagramma trasportata dalla rete).

I campi dell’header IP responsabili della frammentazione sono:

- Flag *MF* (More Fragments)
- Flag *DF* (Don’t Fragment)
- Fragment Offset
- Identification

Flag *DF*:

- A volte la frammentazione può essere controproducente, quindi si può impostare pacchetti con flag *DF* = 1, in questo modo se vengono frammentati generano un errore di *Destination Unreachable* alla sorgente

Flag *MF*:

- Se *MF*=1 Indica che il datagramma è un frammento e non è l’ultimo

Fragment Offset:

- Indica la distanza dal primo frammento del datagramma

Identification:

- Indica il datagramma di cui fa parte.

Possiamo verificare se nella nostra connessione “command” ftp è presente frammentazione IP, applicando il filtro: *“tcp.port == 21 && ip.flags.mf == 1”*.

tcp.port == 21 && ip.flags.mf == 1						
No.	Time	Source	Destination	Protocol	Length	Info

Applicando il filtro notiamo che non ci sono pacchetti frammentati in questa connessione.

Procedo quindi a fare un altro esempio di frammentazione, utilizzando questa volta un pacchetto ICMP più grande della MTU della mia rete.

Utilizzo per questo scopo il comando ***“ping -n 1 -l 5000 “indirizzo ip di un indirizzo sulla mia rete”*** ” che invia un pacchetto icmp(-n 1) di dimensione di 5000 byte(-l 5000).

Utilizzando il filtro *“ip.dst == indirizzo ip del dispositivo a cui eseguire il ping”*, posso facilmente trovare i pacchetti che ho inviato tramite ping a quell’indirizzo.

ip.dst == 10.84.172.250						
No.	Time	Source	Destination	Protocol	Length	Info
27	1.354505	192.168.230.17	10.84.172.250	IPv4		1514 Fragmented IP protocol (proto=I...
28	1.354505	192.168.230.17	10.84.172.250	IPv4		1514 Fragmented IP protocol (proto=I...
29	1.354505	192.168.230.17	10.84.172.250	IPv4		1514 Fragmented IP protocol (proto=I...
30	1.354505	192.168.230.17	10.84.172.250	ICMP		602 Echo (ping) request id=0x0001,...

Cattura con filtro applicato

Posso notare che i 4 pacchetti che appaiono hanno dimensione sommata simile a 5000, ossia alla dimensione del ping che ho inviato, ma per essere sicuro che appartengono tutti e 4 allo stesso datagramma posso esaminare il campo *“Identification”* che è uguale per tutti e 4 i pacchetti.

```

> Frame 27: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bit
> Ethernet II, Src: Intel_fd:d4:8c (b0:a4:60:fd:d4:8c), Dst: a2:41:2f:8f:cl
✓ Internet Protocol Version 4, Src: 192.168.230.17, Dst: 10.84.172.250
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 1500
    Identification: 0x981a (38938) ←
> 001. .... = Flags: 0x1, More fragments
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 128
    Protocol: ICMP (1)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.230.17
    Destination Address: 10.84.172.250
    [Reassembled IPv4 in frame: 30]
    [Stream index: 3]
> Data (1480 bytes)

```

Identification che rappresenta il datagramma che è stato frammentato

Posso inoltre verificare che l'ultimo pacchetto sia effettivamente l'ultimo del datagramma mentre il primo sia effettivamente il primo del datagramma; esaminando in ordine:

1. Che il flag mf sia uguale a 0(per verificare che sia l'ultimo frammento)
2. Che il fragment offset sia uguale a 0(per verificare che sia il primo frammento)

No.	Time	Source	Destination	Protocol
• 27	1.354505	192.168.230.17	10.84.172.250	IPv4
• 28	1.354505	192.168.230.17	10.84.172.250	IPv4
• 29	1.354505	192.168.230.17	10.84.172.250	IPv4
→ 30	1.354505	192.168.230.17	10.84.172.250	ICMP

```

> Frame 30: 602 bytes on wire (4816 bits), 602 bytes captured (4816 bits
> Ethernet II, Src: Intel_fd:d4:8c (b0:a4:60:fd:d4:8c), Dst: a2:41:2f:8f
✓ Internet Protocol Version 4, Src: 192.168.230.17, Dst: 10.84.172.250
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 588
    Identification: 0x981a (38938)
✓ 000. .... = Flags: 0x0
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set ←
    ..0. .... = More fragments: Not set
    ...0 0010 0010 1011 = Fragment Offset: 4440
    Time to Live: 128
    Protocol: ICMP (1)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.230.17
    Destination Address: 10.84.172.250
> [4 IPv4 Fragments (5008 bytes): #27(1480), #28(1480), #29(1480), #3
    [Stream index: 3]

```

Il Flag More Fragment del pacchetto 30 è 0, che conferma sia l'ultimo del datagramma frammentato

ip.dst == 10.84.172.250					
No.	Time	Source	Destination	Protocol	Length
27	1.354505	192.168.230.17	10.84.172.250	IPv4	1500
28	1.354505	192.168.230.17	10.84.172.250	IPv4	1500
29	1.354505	192.168.230.17	10.84.172.250	IPv4	1500
30	1.354505	192.168.230.17	10.84.172.250	ICMP	1500

> Frame 27: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
 > Ethernet II, Src: Intel_fd:d4:8c (b0:a4:60:fd:d4:8c), Dst: a2:41:2f:8f:cd:00 (10:00:00:00:00:00)
 > Internet Protocol Version 4, Src: 192.168.230.17, Dst: 10.84.172.250
 > Transmission Control Protocol, Src Port: 49152, Dst Port: 80, Seq: 38938, Len: 100
 > Hypertext Transfer Protocol, Request Method: GET, Request URI: /, Request Version: 1.1, Status: 200 OK, Content-Type: text/html, Content-Length: 1418, ETag: "1234567890", Last-Modified: Mon, 10 Jun 2013 12:34:56 GMT, Server: Apache/2.4.6-2ubuntu2.17, X-Frame-Options: SAMEORIGIN, X-UA-Compatible: IE=edge, X-XSS-Protection: 1; mode=block, X-Content-Type-Options: nosniff

Il Fragment Offset del pacchetto 27 è uguale a 0, che conferma sia il primo del datagramma.