Progetto Wireshark:

Analisi di un Protocollo TCP-IP

Paolo De Mori
paolo.demori@studio.unibo.it
0001071000

INDICE:

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.

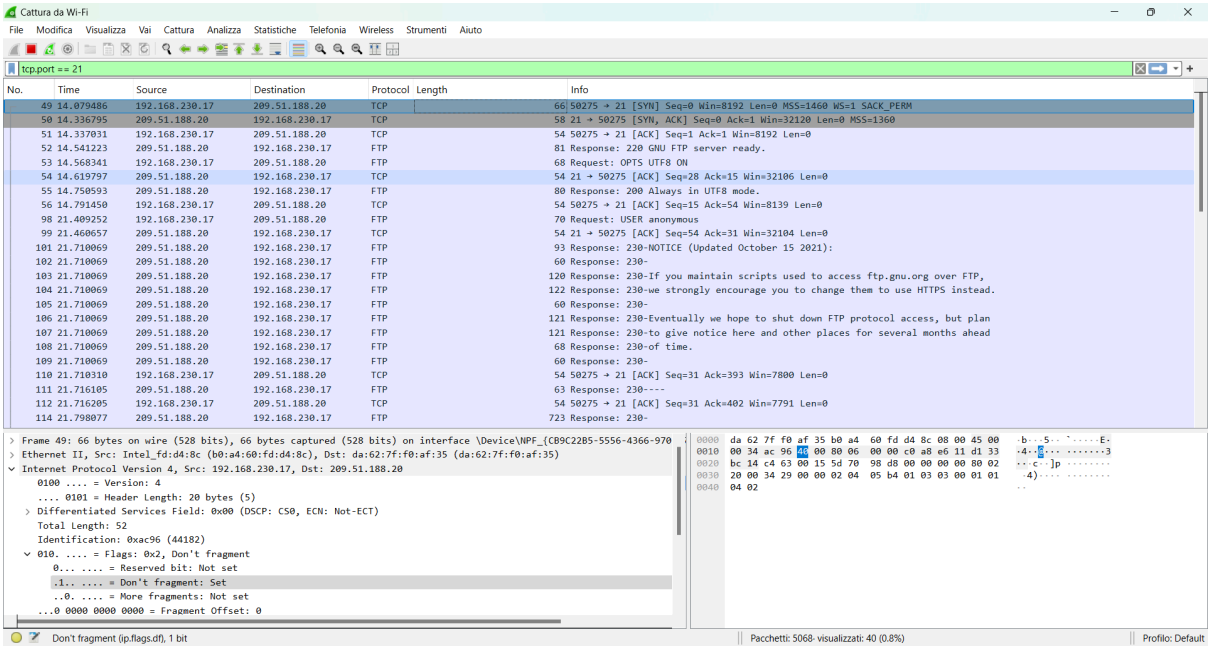
Dopo aver avviato Wireshark mi sono connesso a un server FTP tramite il comando ***ftp*** (fornito nativamente da windows), ho effettuato l'accesso come ***anonymous***, e dopo aver scaricato un file(tramite il comando ***get README***), ho chiuso la connessione tramite il comando ***QUIT***.

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.

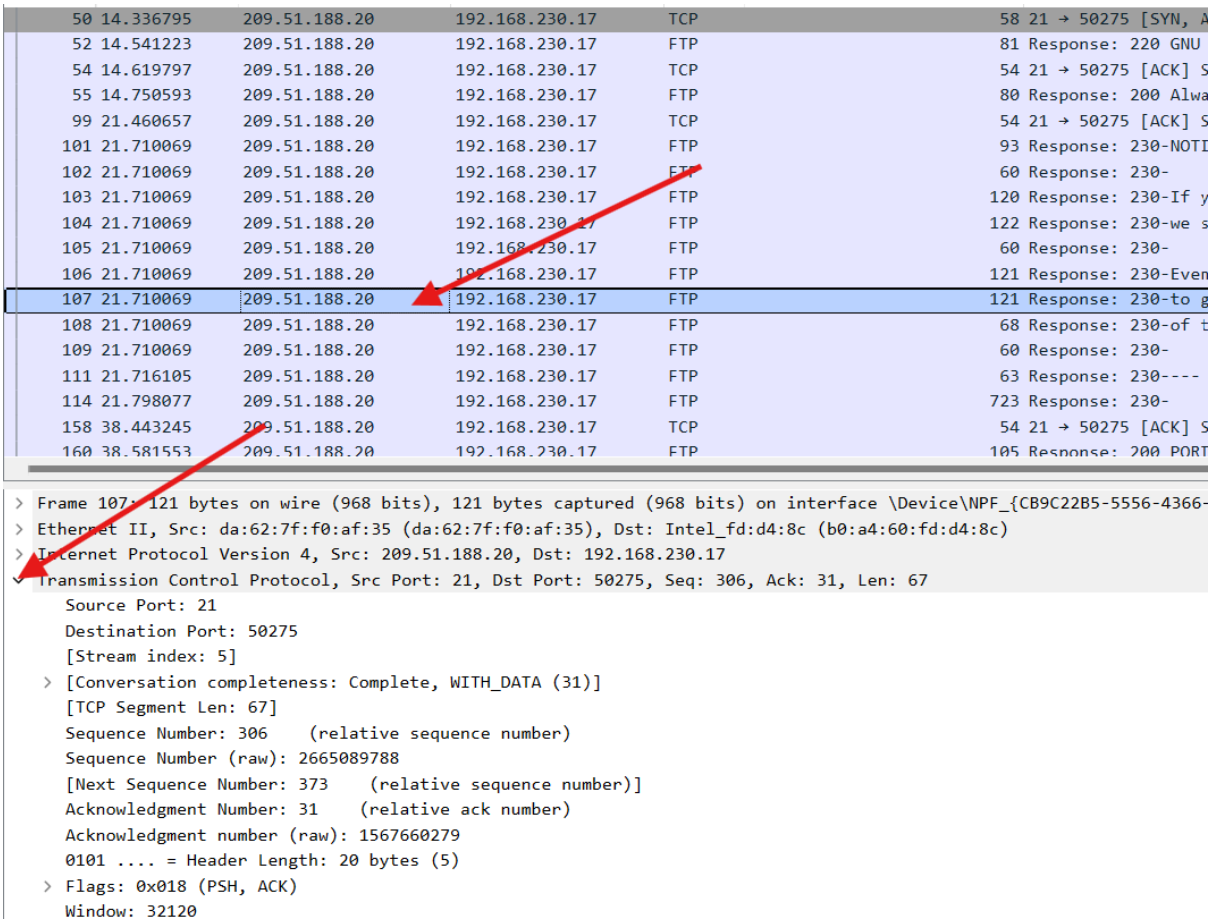


La cattura dei pacchetti filtrati con “tcp.port==21”

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.

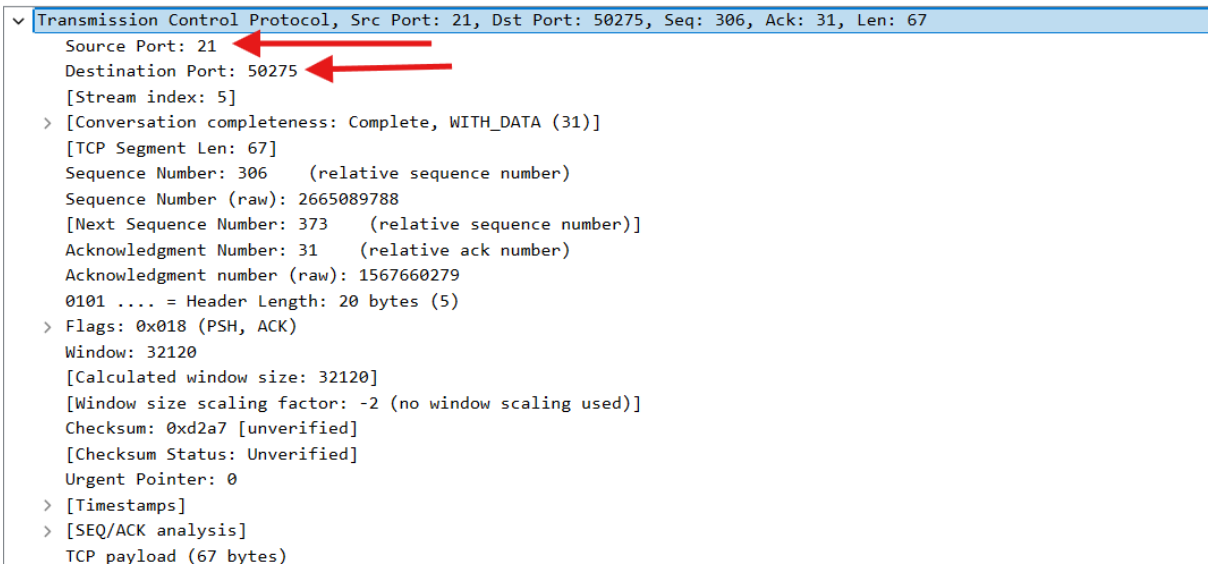
Utilizzando Wireshark, è possibile scegliere un pacchetto(in questo caso il 107) e cliccando nella parte bassa dell'interfaccia sulla voce relativa al *"Transmission Control Protocol"* Wireshark permette di esaminare l'header *"TCP"* e vedere tutti i suo campi:



Come esaminare l'header tcp

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

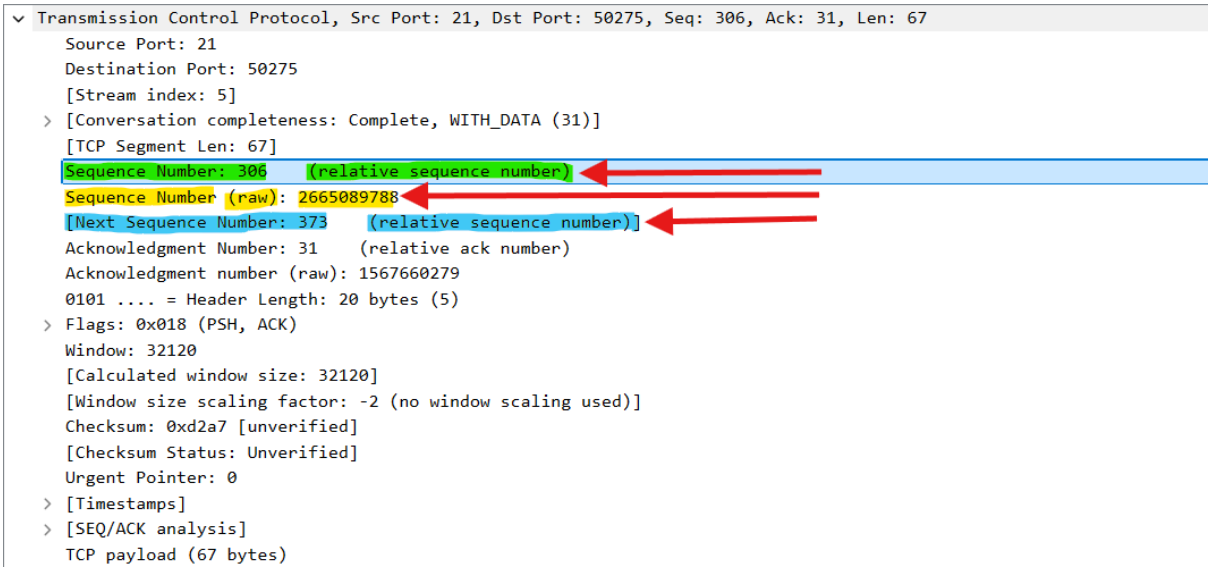
1. *32 bit di Source Port e Destination Port:*



Source Port e Destination Port del pacchetto No.107

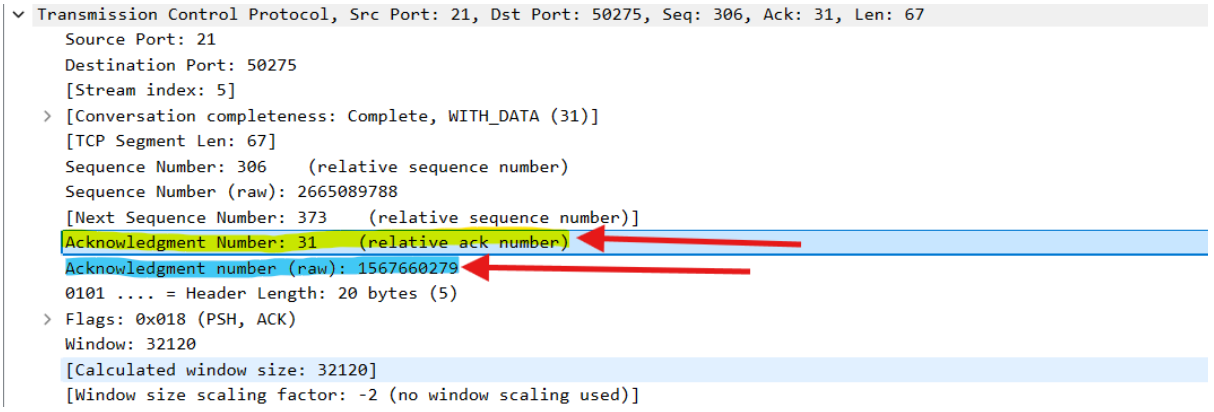
2. *32 bit di sequence number:*
   I dati inviati dai pacchetti tcp sono numerati utilizzando i byte. Il sequence number rappresenta il numero del primo byte dei dati che si stanno inviando. Il primo sequence number di una connessione chiamato *"ISN"* è stabilito al momento dell'instaurazione della connessione 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.



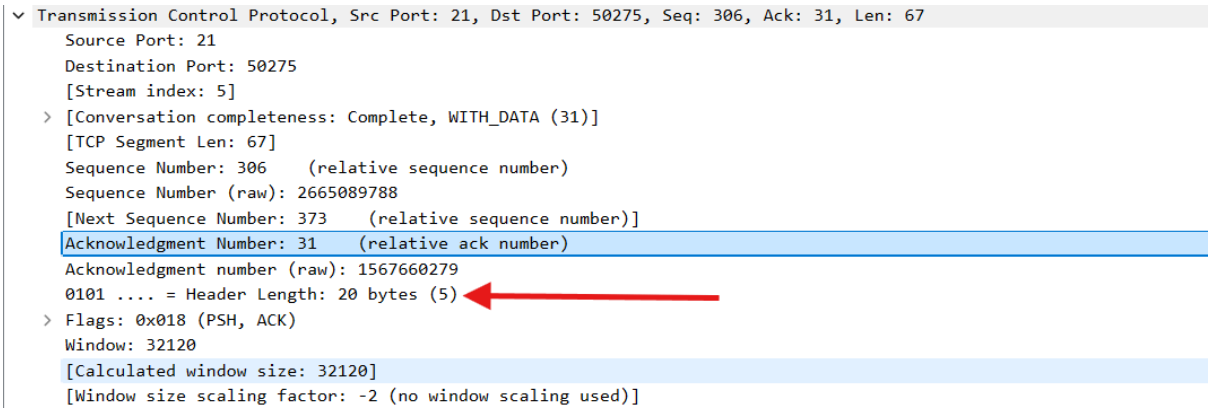
Sequence number del pacchetto No.107

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.



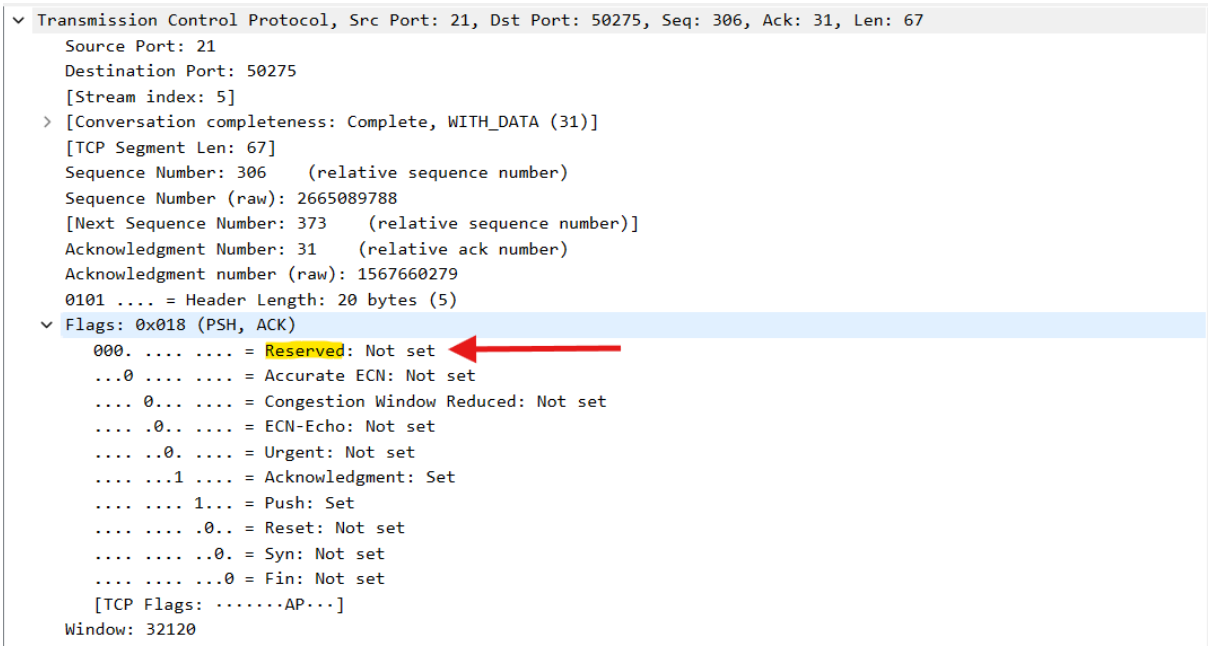
Acknowledgement Number del pacchetto No.107

4. *4 bit di header length:*
   Rappresentano la lunghezza dell'header tcp, variabile in base alla lunghezza del campo *"options"*.



Header length del pacchetto No.107

5. *6 bit reserved:*
   riservati per usi futuri



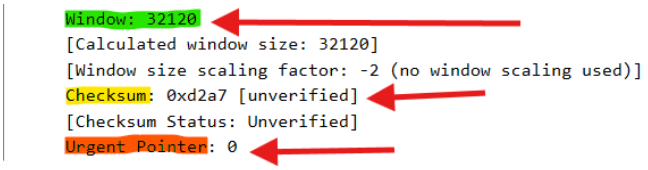
Reserved bits del pacchetto No.107

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···]
```

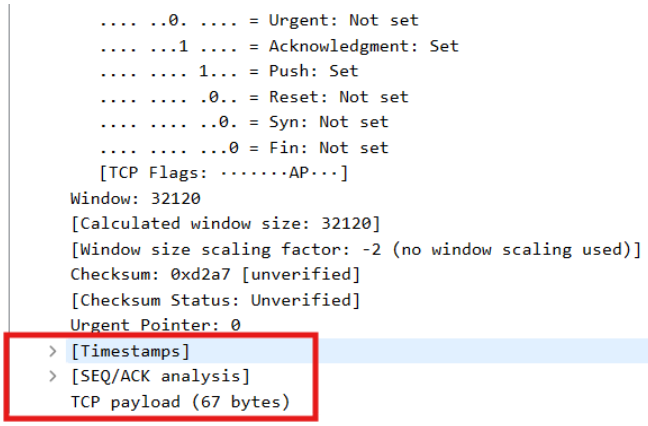
Flags del pacchetto No.107

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 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.



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:

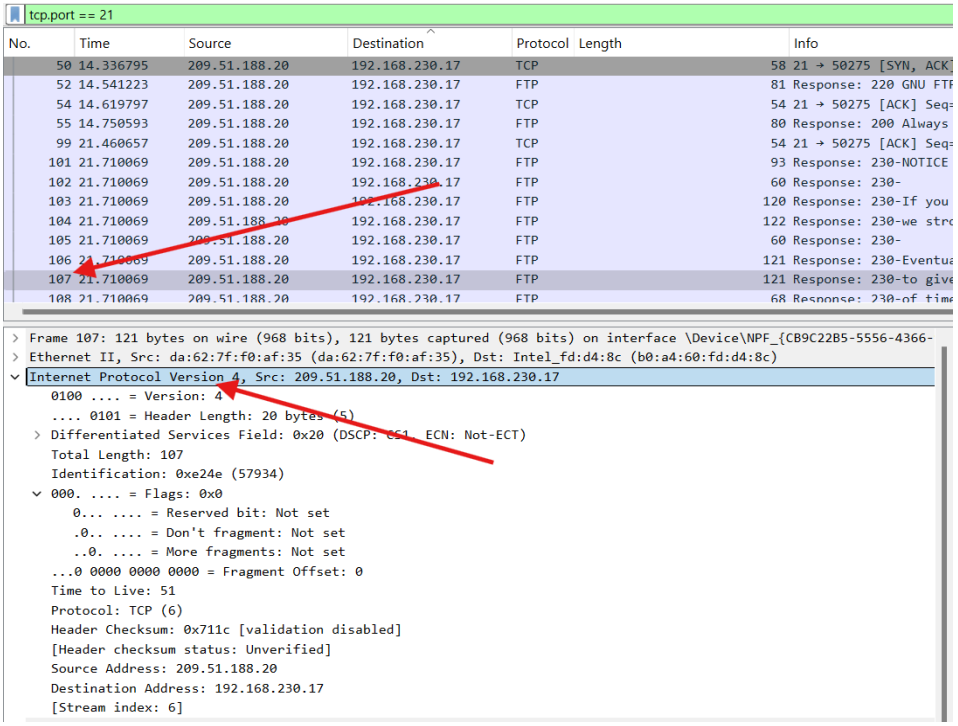
- Ha come protocollo TCP
- Ha come porta sorgente la porta 21

- 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.
E' 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.

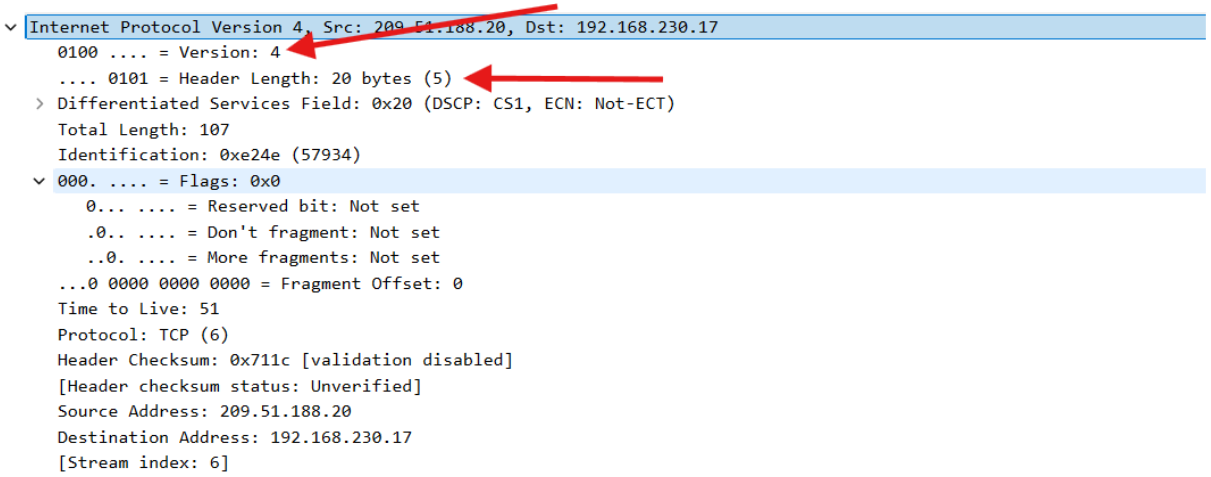


IP Header del datagramma No.107

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

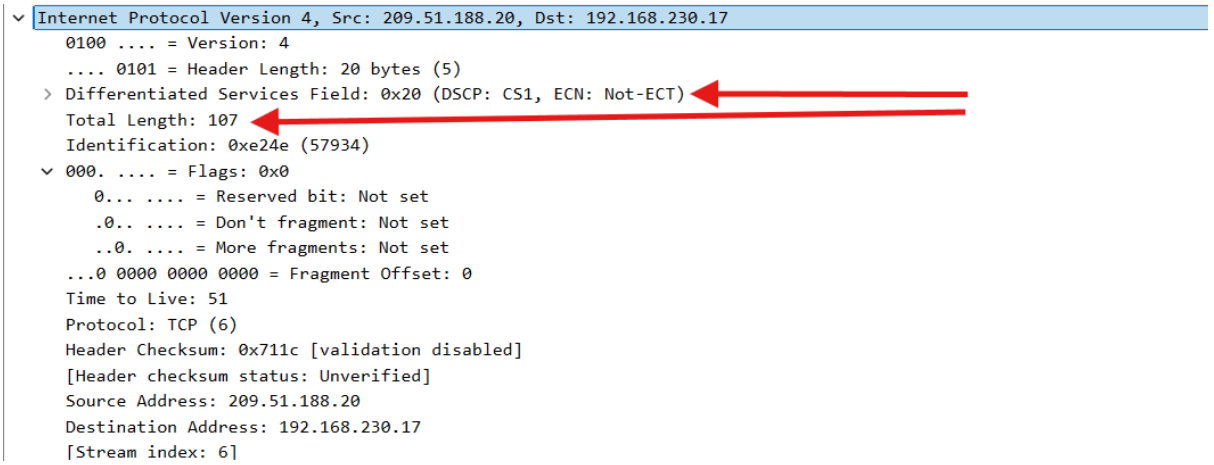
I byte obbligatori sono suddivisi in:

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



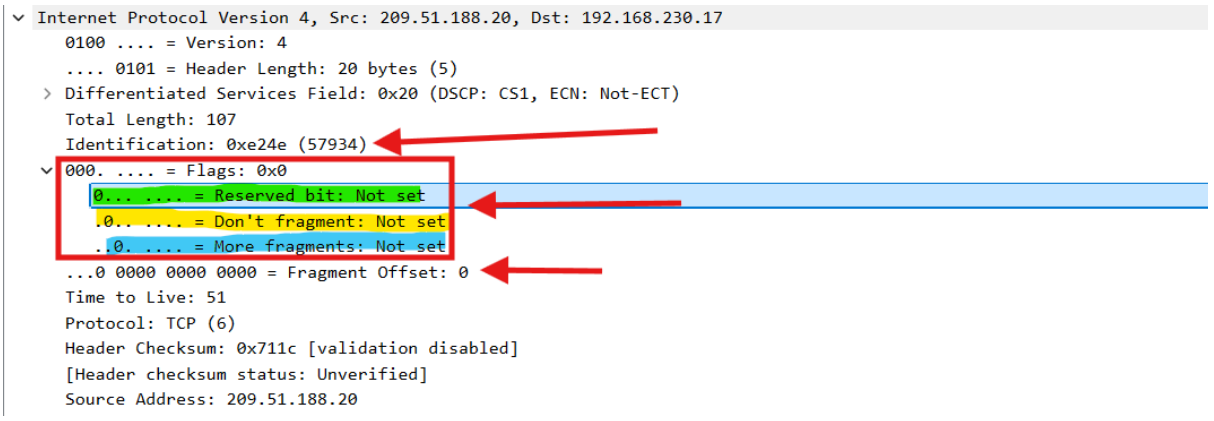
Versione e Header Length del datagramma No.107

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



Tipo di Servizio e Total Length del datagramma No.107

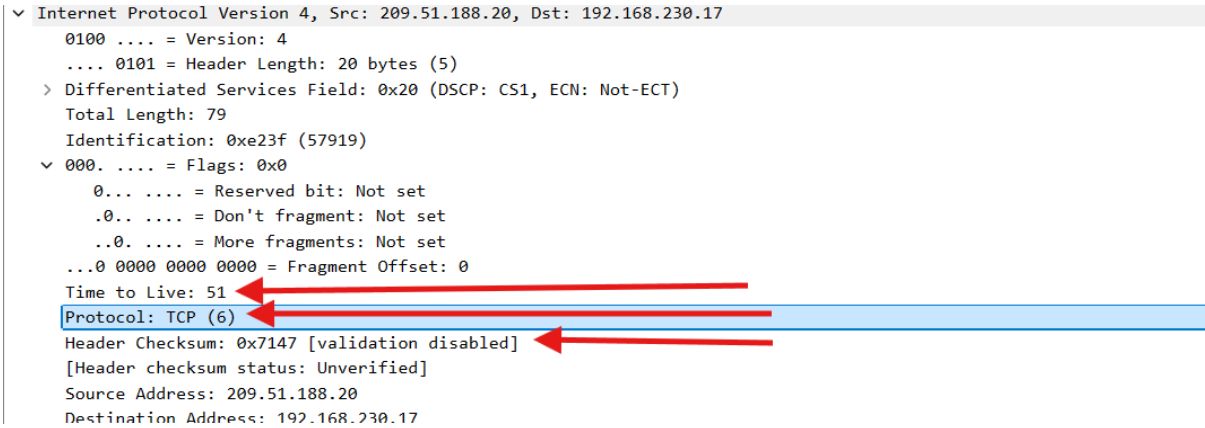
3. *32 bit utilizzati soprattutto per la frammentazione*, suddivisi in:
    - 16 bit di *"Identification":* un numero che rappresenta univocamente il datagramma e permette di identificare a che datagramma corrisponde un certo offset.
    - 3 bit per le flags rispettivamente:
        - 1 bit sempre a 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)



Flags del datagramma No.107

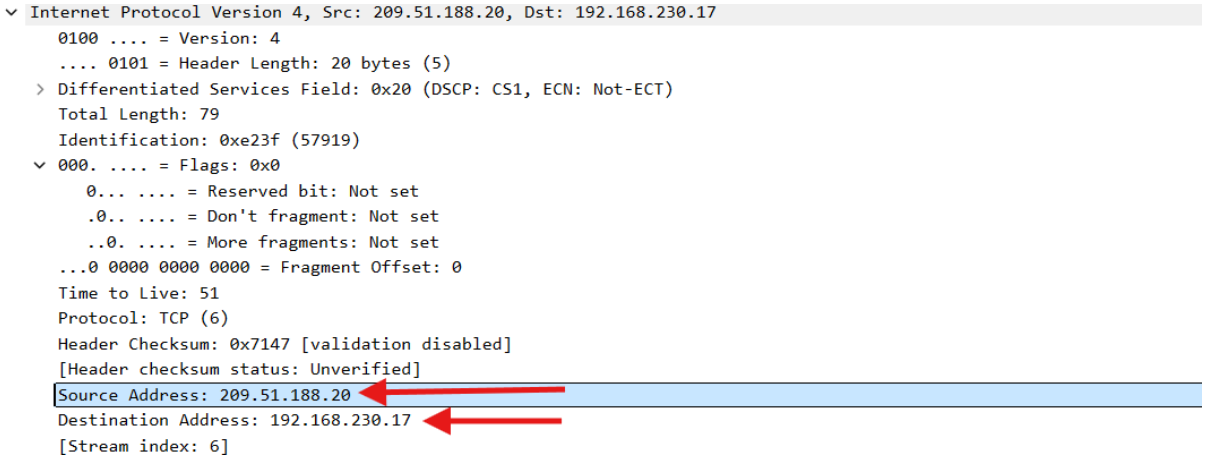
4.
    a. *8 Bit per il "time to live"(TTL)*, ossia il massimo numero di nodi attraversabili dal pacchetto. Tale valore viene decrementato da ogni nodo che attraversa, una volta a 0 il pacchetto viene distrutto
    b. *8 Bit per rappresentare il protocollo di livello superiore che utilizza IP*

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



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:*



Source Address e Destination Address del datagramma No.107

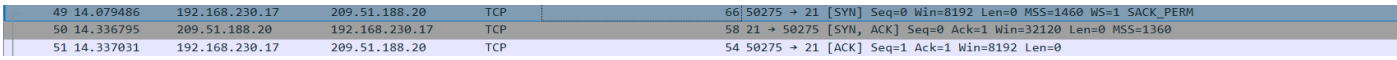
Resoconto del pacchetto in analisi:

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.

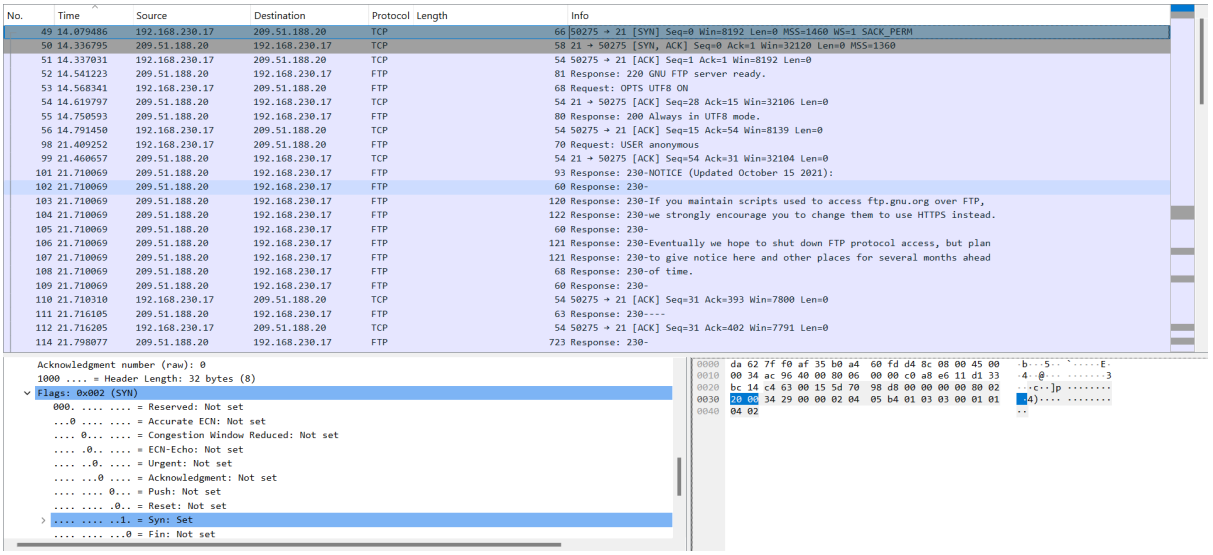
Three Ways Handshake:

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 *"l' acknowledgement number"*(con la sigla Ack).



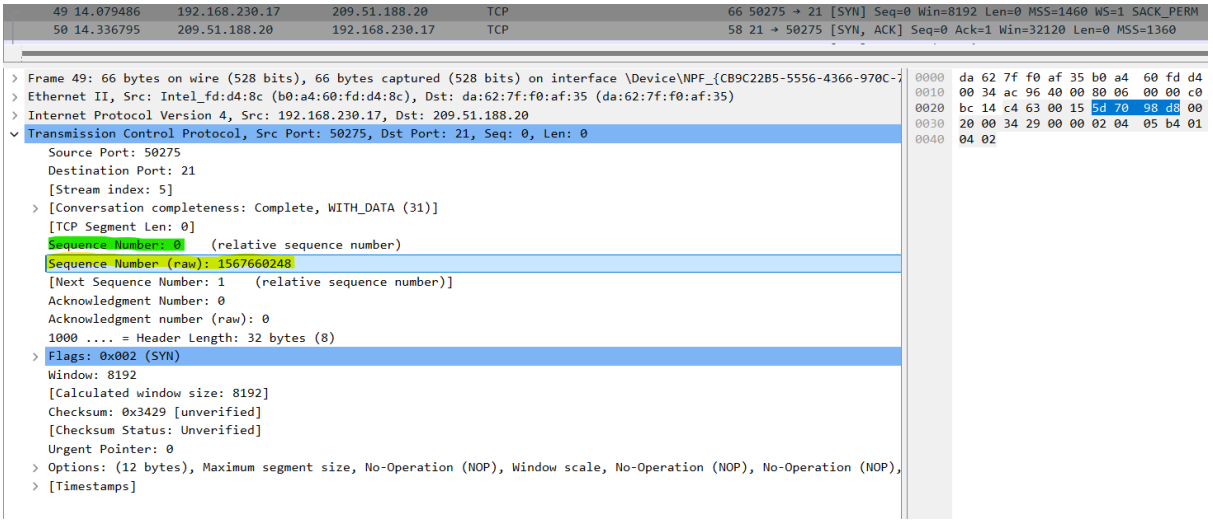
i Tre pacchetti della "three ways handshake"

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.



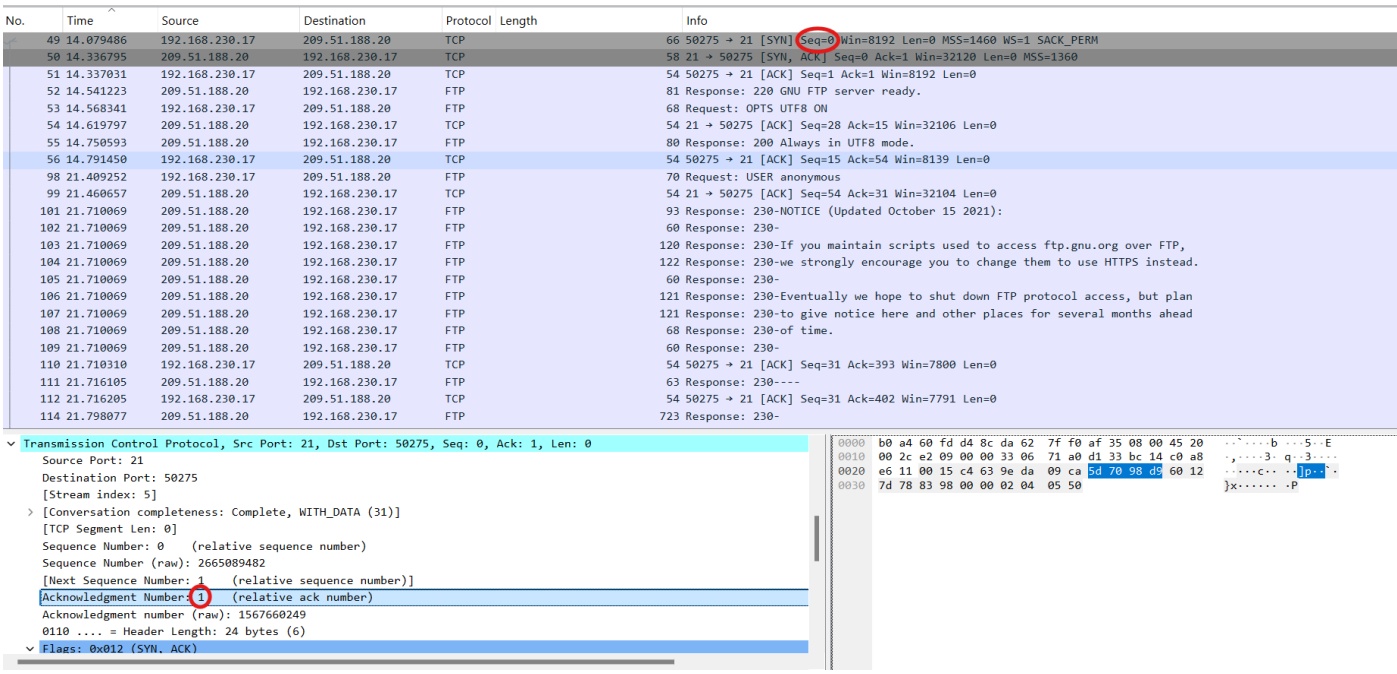
analisi dei flag del primo pacchetto

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.

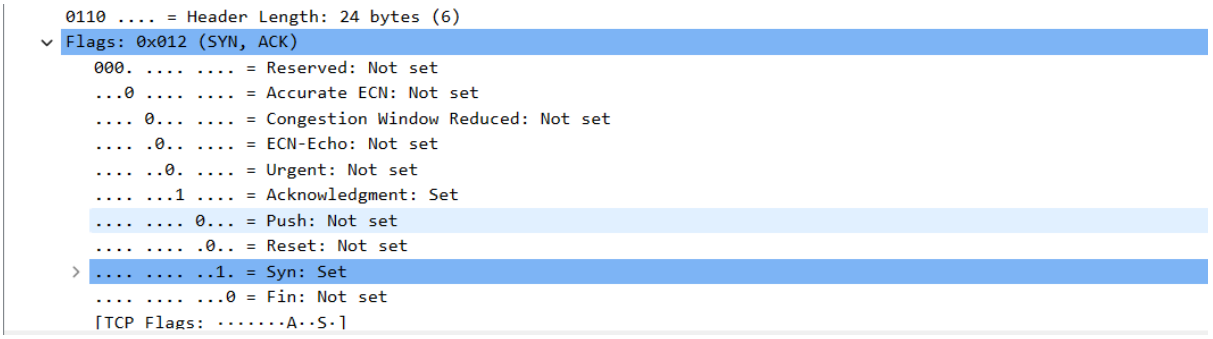


ISN

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.

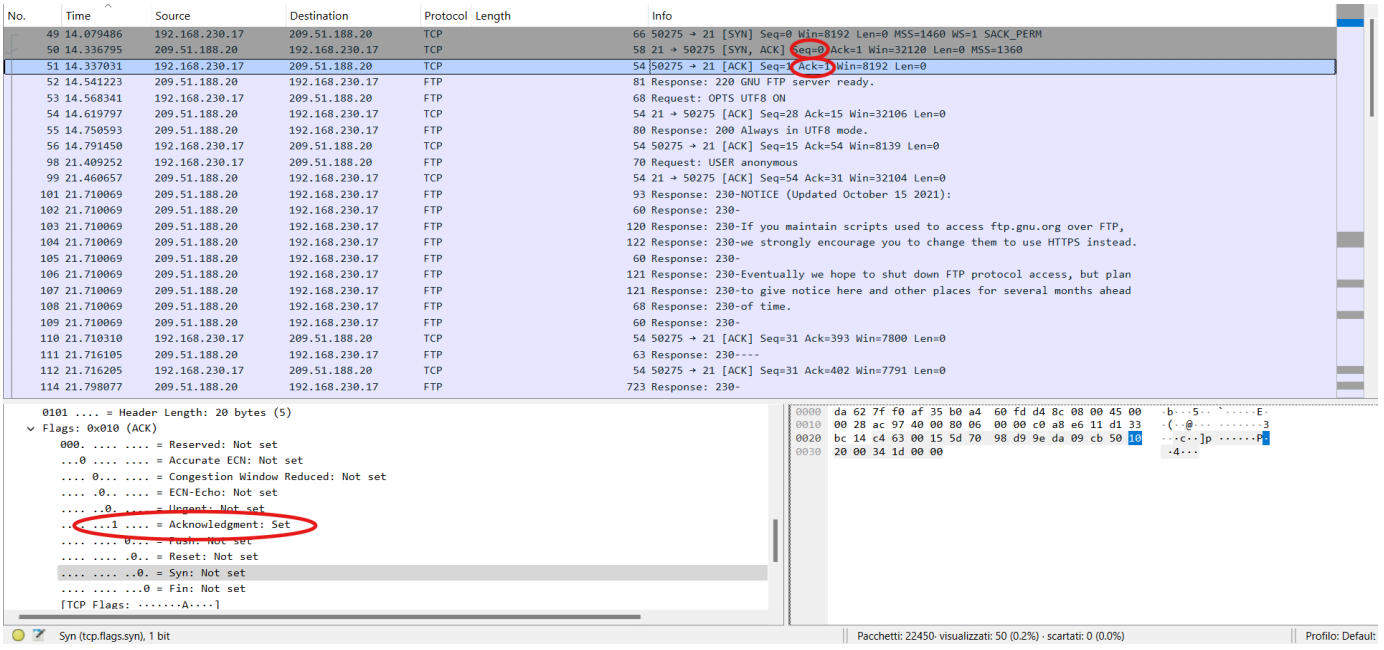


Acknowledge Number del secondo pacchetto pari al sequence number +1 del primo

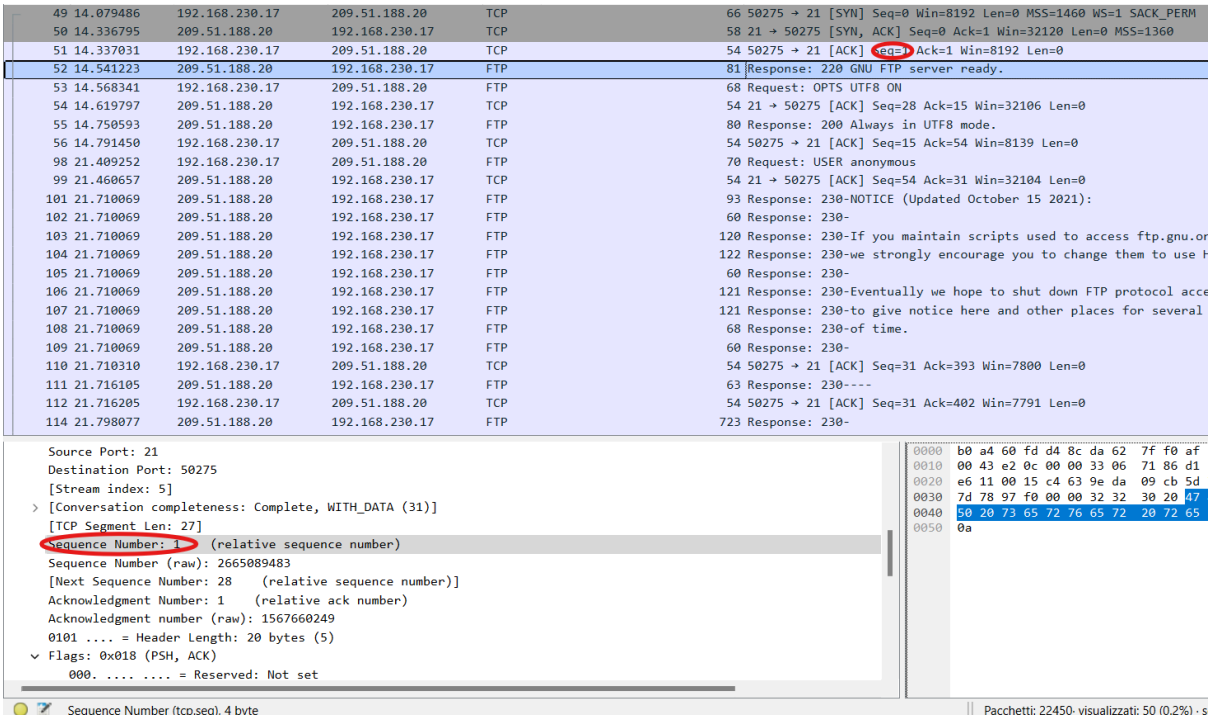


Analisi dei flag del secondo pacchetto

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.



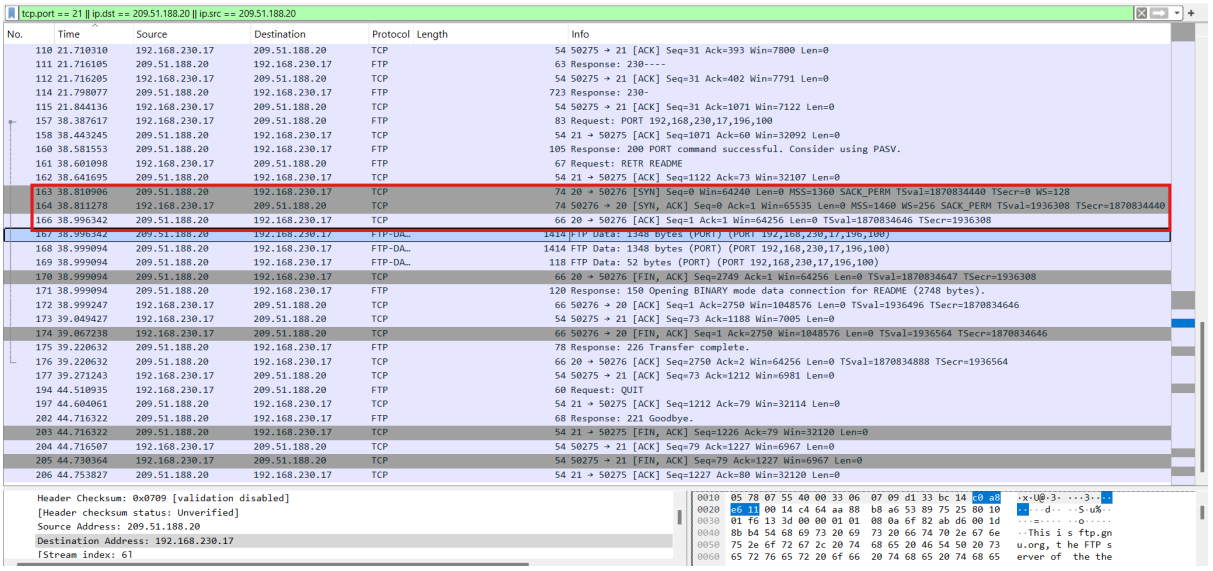
Acknowledgement number del terzo pacchetto uguale al sequence number del primo più 1



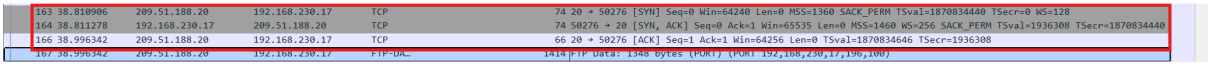
Sequence number del terzo pacchetto è uguale al primo pacchetto contenente dati

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, potrò facilmente notare un'altra three ways handshake utilizzata per creare la connessione *"data"*.

Il filtro per visualizzare anche la connessione "data"



La three ways handshake della connessione “data”



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’inio 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.

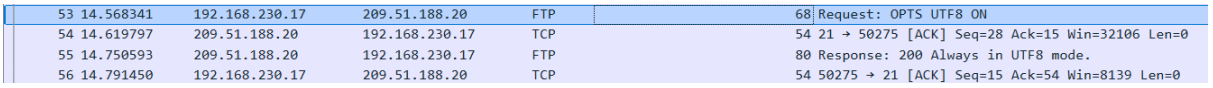
Acknowledgement Number:

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”*.
“l’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:

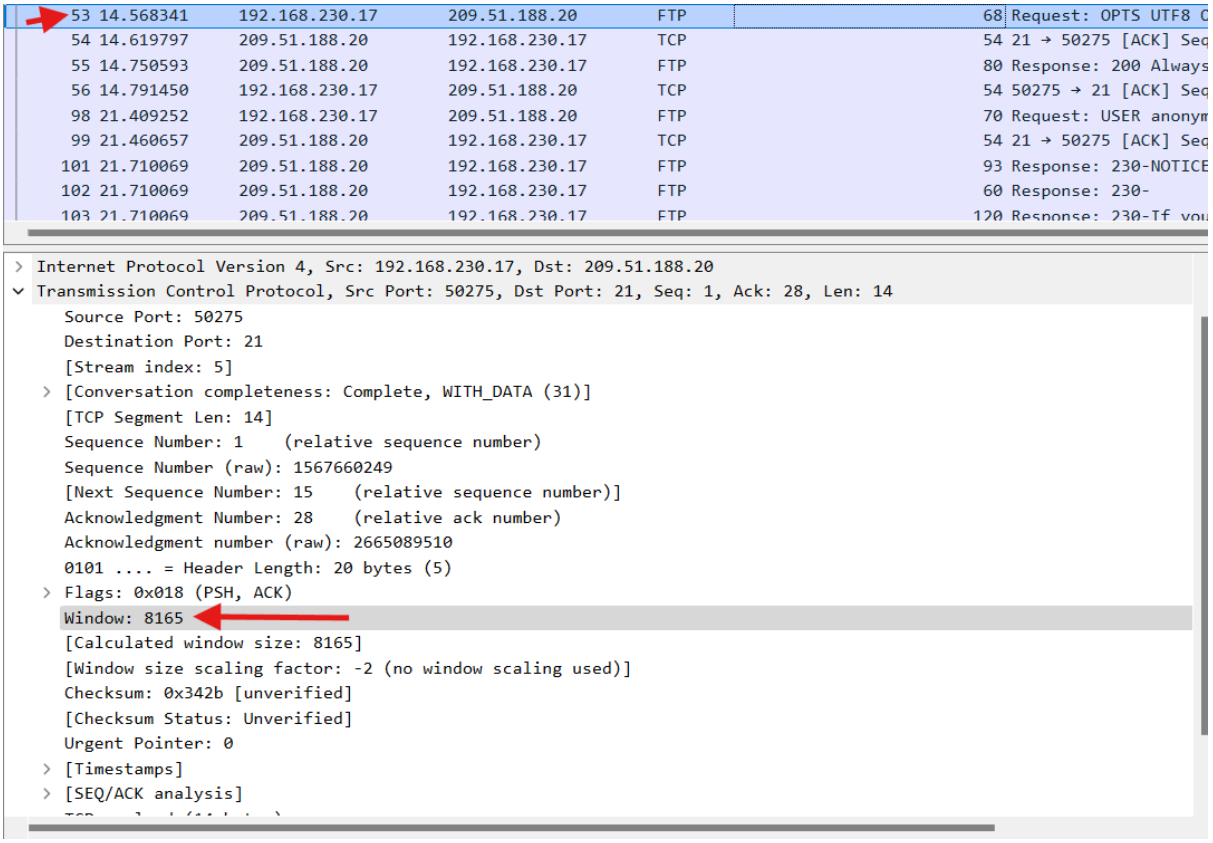
1. Se già ricevuto, lo scarta
2. Se mancano uno o più segmenti, se il buffer lo permette, il ricevente memorizza il segmento e ri-invia *“l’ acknowledgement”* dell’ultimo segmento in ordine. In caso riceva i pacchetti mancanti, invierà le conferme necessarie, se invece scade il *“persist timer”*, senza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

Analisi del controllo di flusso nella cattura effettuata:

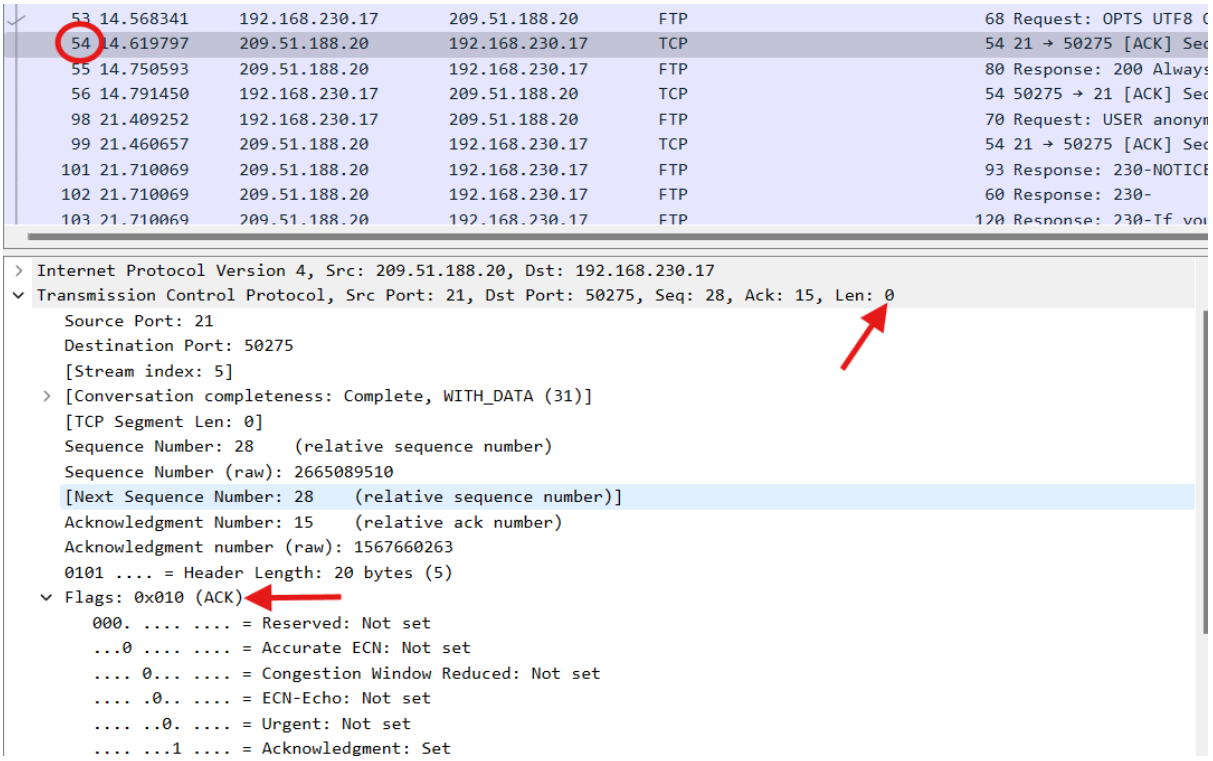
Nella nostra cattura è possibile identificare facilmente il meccanismo di controllo di flusso TCP.
Prendiamo per esempio in analisi i pacchetti No. 53,54, 55 e 56



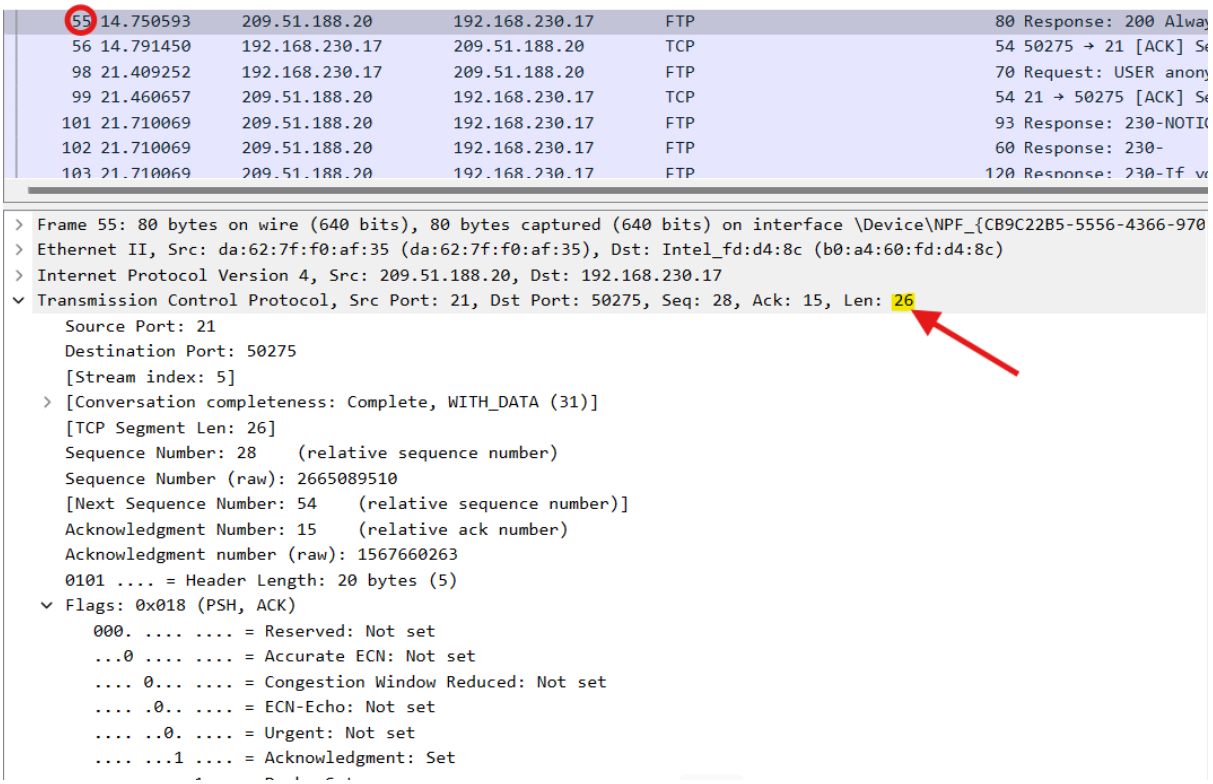
Possiamo verificare che al momento dell’invio del pacchetto 53, il mittente”192.168.230.17” ha come window size “8165”.



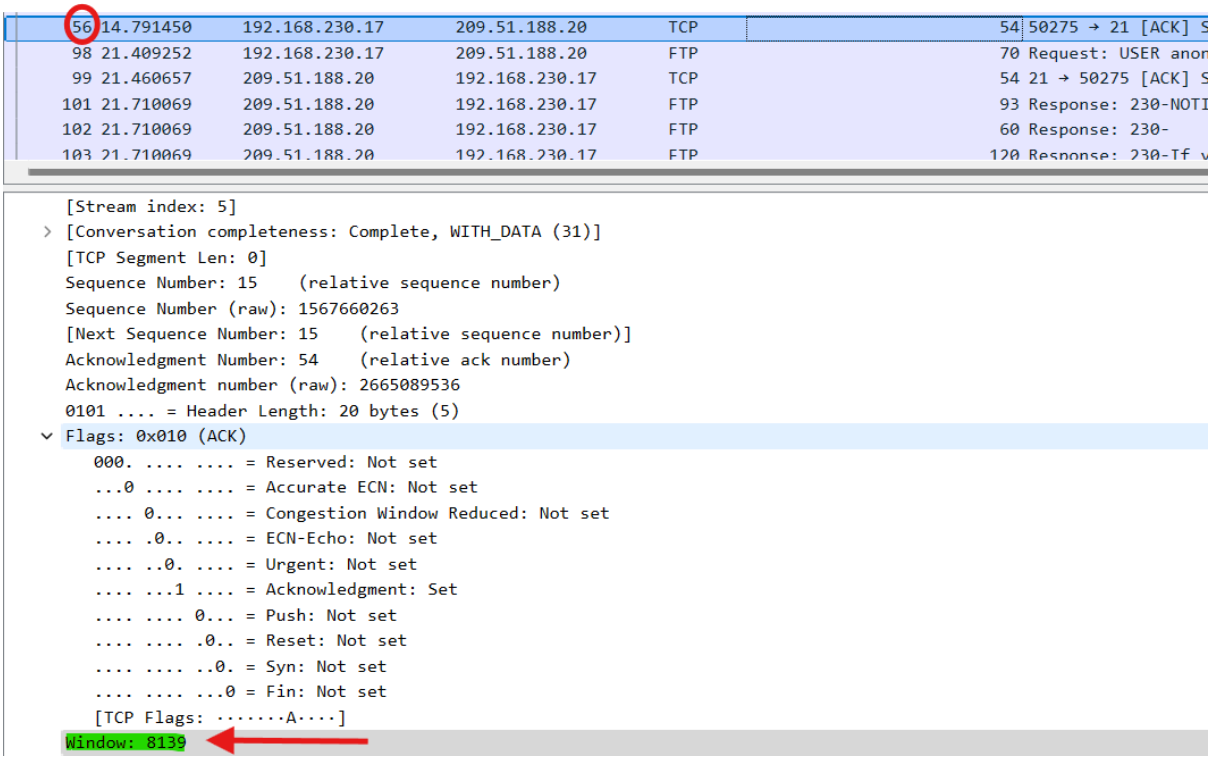
Successivamente possiamo notare come l'interlocutore "209.51.188.20" invia due pacchetti.
Il pacchetto 54 è un pacchetto di *acknowledgement* puro, quindi con 0 byte di dati trasportati(ovviamente escluso header).



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

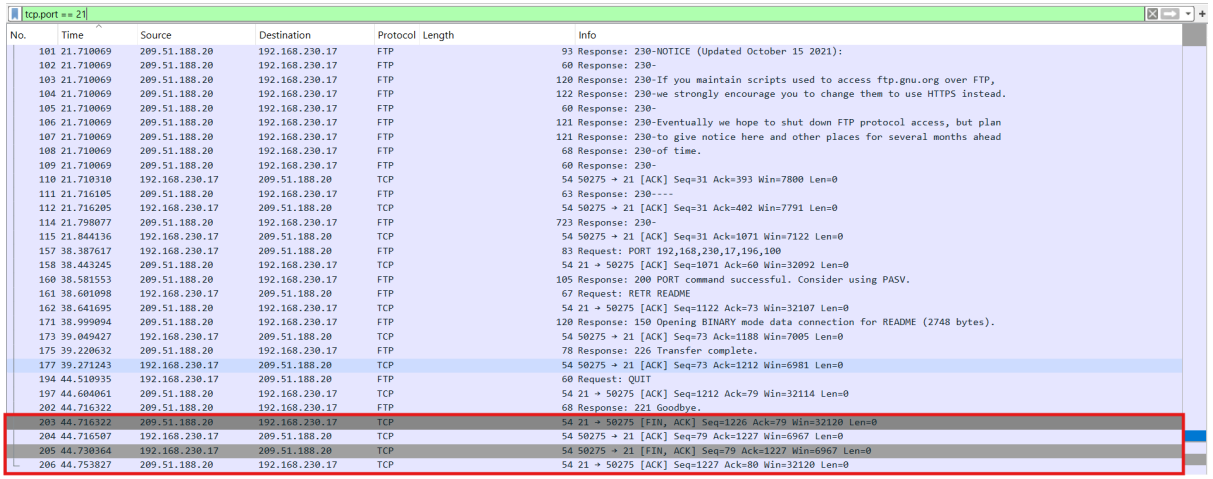


Quindi possiamo aspettarci che la grandezza del buffer del ricevitore prima dell'invio di un ack sia 8165 -(26+0), quindi 8139.
Effettivamente controllando il pacchetto numero 56, verifichiamo la correttezza della nostra ipotesi.



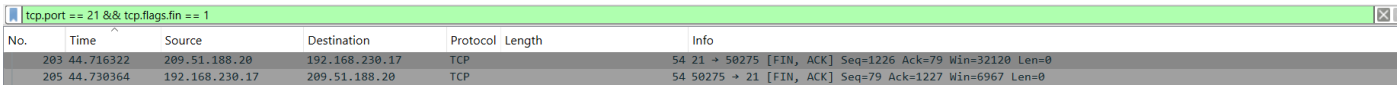
Chiusura della Connessione:

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.



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.



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

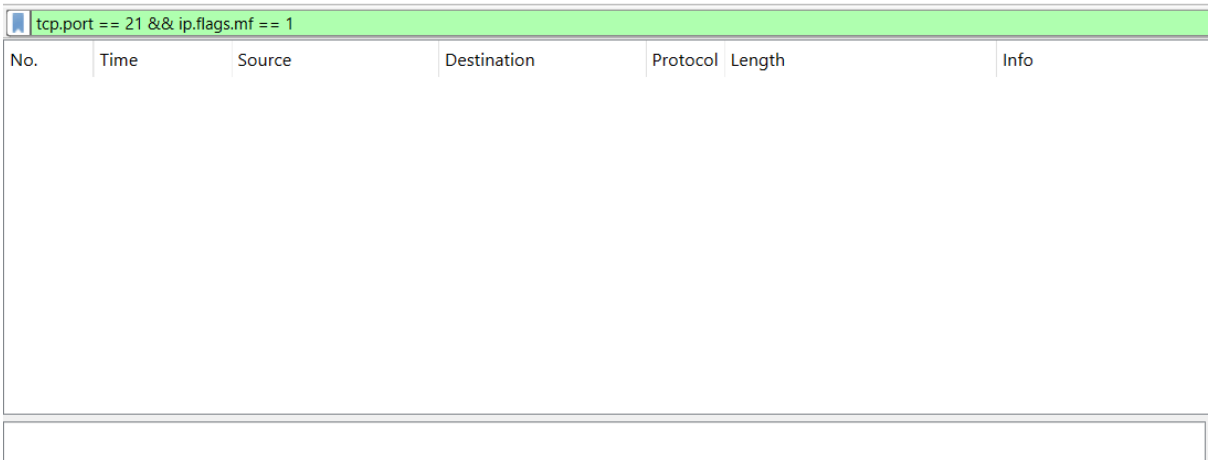
Fragment Offset:

- Indica la distanza dal primo frammento del datagramma

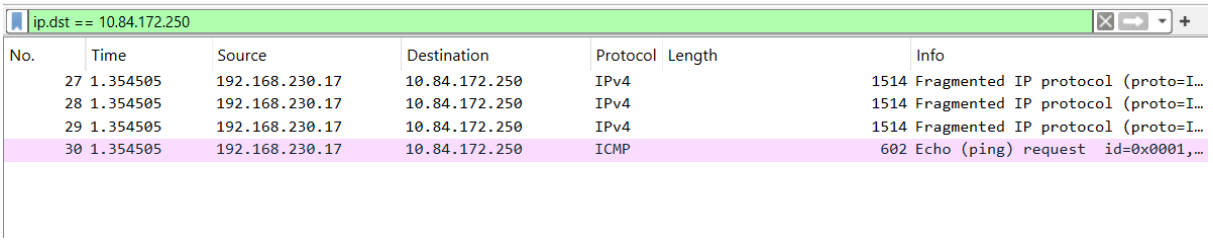
Identification:

- 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”.*

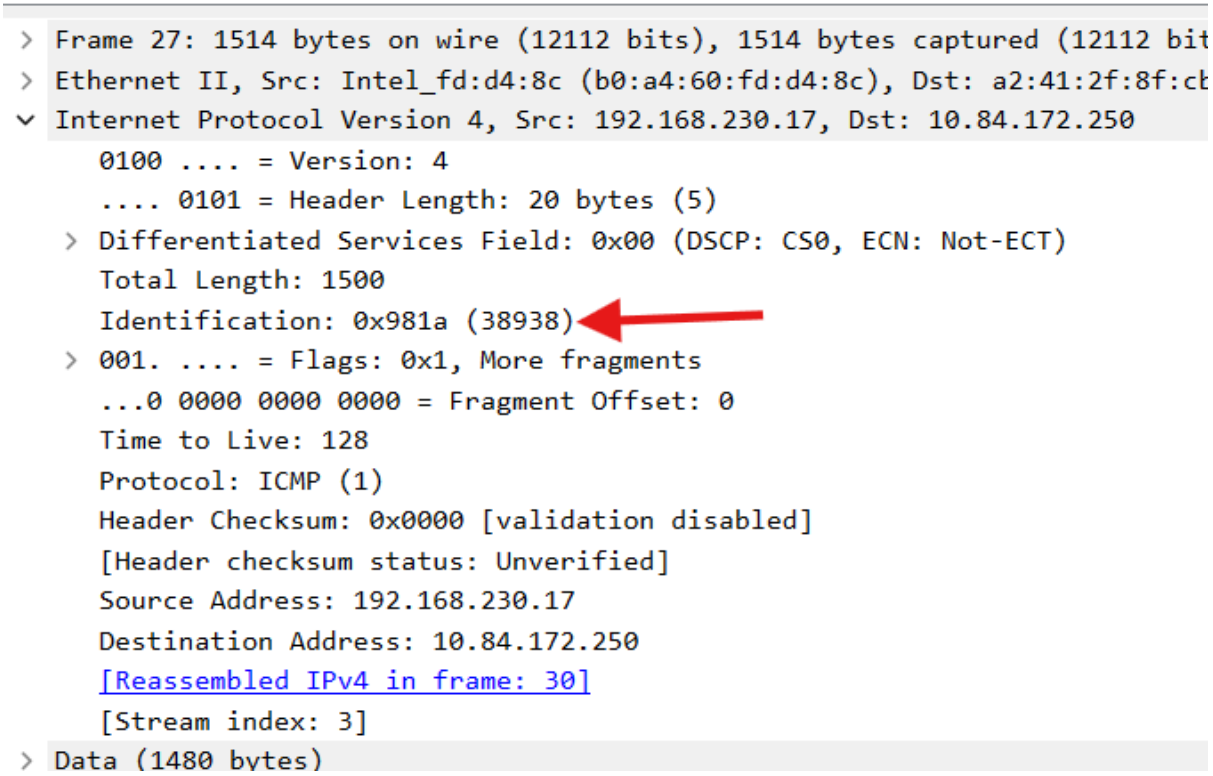


Applicando il filtro notiamo che non ci sono pacchetti frammentati in questa connessione.
Procedo quindi a fare un altro esempio di frammentazione, utilizzando questa volta un pacchetto ICMP più grande della MTU della mia rete.
Utilizzo per questo scopo il comando *“**ping -n 1 -l 5000 “indirizzo ip 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.



Cattura con filtro applicato

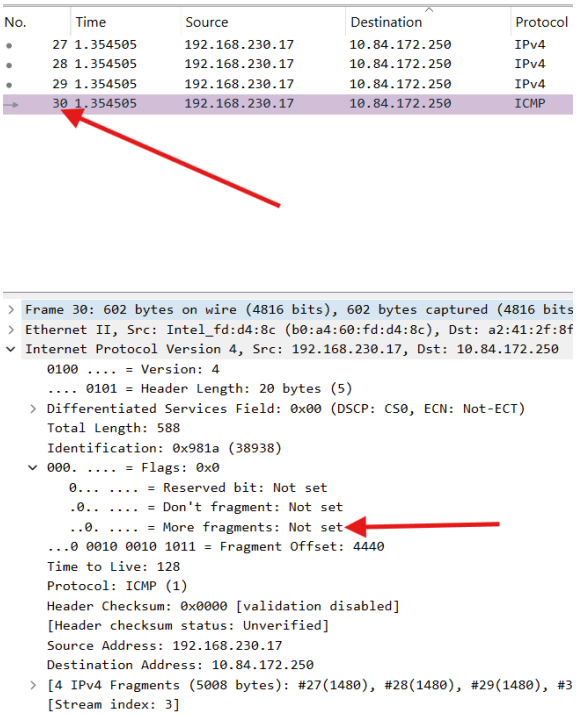
Posso notare che i 4 pacchetti che appaiono hanno dimensione sommata simile a 5000, ossia alla dimensione del ping che ho inviato, ma per essere sicuro che appartengono tutti e 4 allo stesso datagramma posso esaminare il campo *“Identification”* che è uguale per tutti e 4 i pacchetti.



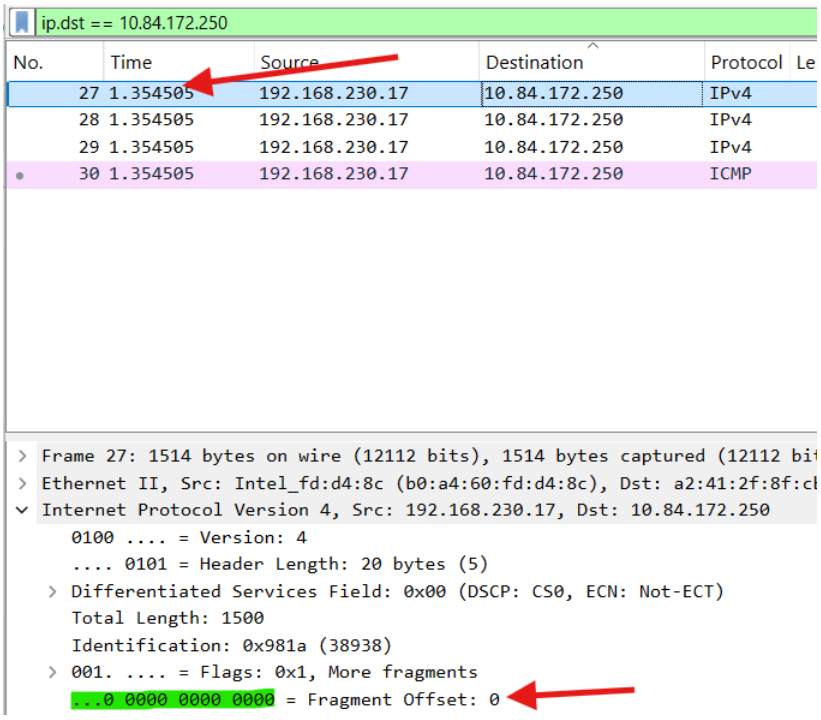
Identification che rappresenta il datagramma che è stato frammentato

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)



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



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