Consegna del progetto

Progetto Wireshark: Analisi del Protocollo TCP/IP

Descrizione: Usare Wireshark per catturare e analizzare il traffico di rete durante una sessione di trasferimento file (es. FTP o HTTP). Gli studenti dovranno esaminare i pacchetti per comprendere il funzionamento del protocollo TCP/IP.

Obiettivi: Identificare la sequenza di handshake TCP, analizzare la frammentazione dei pacchetti e il controllo di flusso, e riconoscere i campi chiave nei pacchetti.

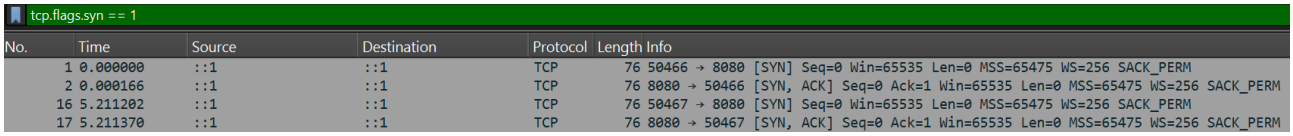
Consegne richieste: Report con screenshot delle catture di pacchetti, analisi delle metriche chiave (es. numero di pacchetti, tempi di latenza) e spiegazione di eventuali ritrasmissioni o problemi di connessione osservati.

SETUP:

Ho creato un server con il comando: **python3 -m http.server 8080**, in modo da poter scaricare un qualsiasi file, per analizzarlo e catturare il traffico di loopback. Nel mio caso ho creato un file "programma da scaricare.txt" e dopo aver avviato la cattura su Wireshark, ho proceduto a scaricare il file entrando tramite browser nella pagina http://localhost:8080/.

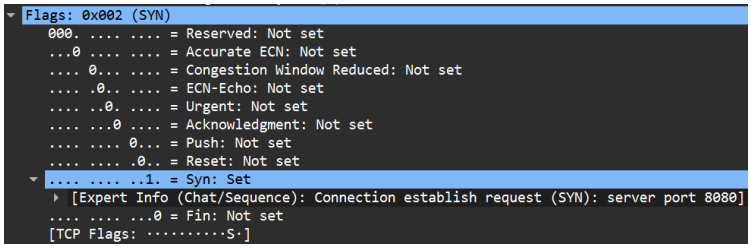
Ho fatto un ulteriore esempio andando a scaricare XAAMP tramite browser per andare ad analizzare la perdita di pacchetti, catturando il traffico su WI-FI.

SEQUENZA DI HANDSHAKE:

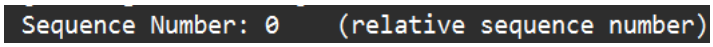


Filtro i pacchetti con "tcp.flags.syn == 1" per trovare la sequenza di handshake TCP(solamente SYN e SYN-ACK, poi imposto "tcp.flags.ack == 1 && tcp.flags.syn == 0").

SYN: pacchetto iniziale inviato dal client(50466) al server(8080).

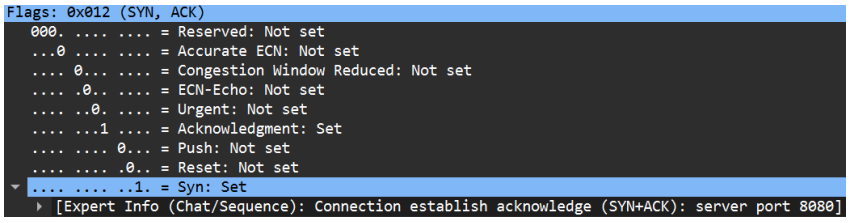


Flags SYN (0x002): il flag indica che questo pacchetto avvia la connessione.



Sequence Number 0: indica che il numero iniziale di sequenza per il flusso di dati.

SYN-ACK: risposta del server che conferma di aver ricevuto SYN per l'accettazione della connessione.



Acknowledgment Number 1: Il server conferma di aver ricevuto il SYN del client, indicando il prossimo numero di sequenza atteso.

ACK: il client risponde con un ACK, completando l'handshake.

```
Sequence Number: 1    (relative sequence number)
Sequence Number (raw): 759322093
[Next Sequence Number: 1    (relative sequence number)]
Acknowledgment Number: 1    (relative ack number)
Acknowledgment number (raw): 1979749798
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····]
```

Sequence number 1: client invia il proprio numero di sequenza successivo al SYN.
Acknowledgment number 1: client conferma di aver ricevuto SYN-ACK.

TRASFERIMENTO HTTP:

http

No.	Time	Source	Destination	Protocol	Length	Info
4	0.000552	::1	::1	HTTP	713	GET / HTTP/1.1
8	0.009465	::1	::1	HTTP	3629	HTTP/1.0 200 OK (text/html)
19	5.211786	::1	::1	HTTP	475	GET /programma%20da%20scaricare.txt HTTP/1.1
23	5.214610	::1	::1	HTTP	6112	HTTP/1.0 200 OK (text/plain)

Pacchetto 4: richiesta HTTP GET dal client al server per la risorsa / (nel mio computer).
Pacchetto 8: risposta del server alla richiesta.
Pacchetto 19: Richiesta HTTP GET per andare a scaricare il file /programma da scaricate.txt dal server localhost:8080.

```
[Timestamps]
   [Time since first frame in this TCP stream: 0.000584000 seconds]
   [Time since previous frame in this TCP stream: 0.000338000 seconds]
```

Tempo dall'inizio della connessione e tempo dall'ultimo pacchetto.
Pacchetto 23: il server risponde al GET del client con un pacchetto http 200 OK, inviando il file .txt.

```
▾ [Timestamps]
     [Time since first frame in this TCP stream: 0.003408000 seconds]
     [Time since previous frame in this TCP stream: 0.000657000 seconds]
```

Latenza tra richiesta e risposta: 0.003408 - 0.000584 = 0.002824 seconds (2.8 ms)
Non è stata rilevata frammentazione dei pacchetti, ritrasmissioni o problemi di connessione durante il trasferimento HTTP. Il download è avvenuto senza perdita di pacchetti, con una latenza minima di 2.8 ms e un trasferimento completo dei dati.

ESEMPIO SCARICANDO XAAMP

FRAMMENTAZIONE DEI PACCHETTI:

La frammentazione si verifica quando i pacchetti superano la dimensione massima dell'MTU (Maximum Transmission Unit) e vengono divisi in più frammenti.
Frammentazione pacchetti controllata con ip.flags.mf == 1 || ip.frag_offset > 0, nulla da segnalare.

CONTROLLO DELLE RITRASMISSIONI

Uso "**tcp.analysis.retransmission**" per mostrare i pacchetti ritrasmessi(sono copie di pacchetti precedenti inviate per compensare perdite/timeout).
Durante il trasferimento, sono state osservate ritrasmissioni TCP tra il server (87.121.121.2) e il client (192.168.1.15).

```
1506 [TCP Fast Retransmission] 443 → 51093 [ACK] Seq=4437 Ack=2833 Win=64128 Len=1452 [TCP PDU reassembled in 297]
1506 [TCP Retransmission] 443 → 51093 [ACK] Seq=11697 Ack=2833 Win=64128 Len=1452 [TCP PDU reassembled in 297]
1506 [TCP Retransmission] 443 → 51093 [ACK] Seq=17505 Ack=2833 Win=64128 Len=1452 [TCP PDU reassembled in 297]
1506 [TCP Retransmission] 443 → 51093 [ACK] Seq=18957 Ack=2833 Win=64128 Len=1452
1506 [TCP Fast Retransmission] 443 → 51093 [ACK] Seq=8793 Ack=2833 Win=64128 Len=1452 [TCP PDU reassembled in 297]
```

Dimensione media dei pacchetti ritrasmessi: 1452 bytes.
Nonostante le ritrasmissioni, il trasferimento è stato completato correttamente.
Le ritrasmissioni avvenute sono 94, tra Fast Retrasmission e Retrasmission.
Le ritrasmissioni sono causate da:

- Perdita di pacchetti: Il destinatario non ha ricevuto il pacchetto originale.

Con **tcp.analysis.duplicate_ack** vediamo gli ACK duplicati che segnalano al mittente che sta aspettando un pacchetto specifico(dove il Sequence Number è uguale in più pacchetti).

```
268 12.312644   192.168.1.15   87.121.121.2   TCP   66 [TCP Dup ACK 266#1] 51093 → 443 [ACK] Seq=2833 Ack=4437 Win=65280 Len=0 SLE=13149 SRE=14601
271 12.379826   192.168.1.15   87.121.121.2   TCP   66 [TCP Dup ACK 266#2] 51093 → 443 [ACK] Seq=2833 Ack=4437 Win=65280 Len=0 SLE=13149 SRE=16053
272 12.380054   192.168.1.15   87.121.121.2   TCP   66 [TCP Dup ACK 266#3] 51093 → 443 [ACK] Seq=2833 Ack=4437 Win=65280 Len=0 SLE=13149 SRE=17505
275 12.380265   192.168.1.15   87.121.121.2   TCP   74 [TCP Dup ACK 266#4] 51093 → 443 [ACK] Seq=2833 Ack=4437 Win=65280 Len=0 SLE=20409 SRE=23313
284 12.466446   192.168.1.15   87.121.121.2   TCP   74 [TCP Dup ACK 281#1] 51093 → 443 [ACK] Seq=2833 Ack=8793 Win=65280 Len=0 SLE=11697 SRE=17505
285 12.466516   192.168.1.15   87.121.121.2   TCP   74 [TCP Dup ACK 281#2] 51093 → 443 [ACK] Seq=2833 Ack=8793 Win=65280 Len=0 SLE=11697 SRE=18957
```

- Timeout: Il mittente non ha ricevuto un ACK per il pacchetto entro un certo intervallo di tempo.

```
▾ [Timestamps]
    [Time since first frame in this TCP stream: 12.655948000 seconds]
    [Time since previous frame in this TCP stream: 0.000000000 seconds]
```

- Congestione della rete: Ritardi dovuti a traffico intenso.

tcp.analysis.out_of_order

	Time	Source	Destination	Protocol	Length	Info
278	12.447514	87.121.121.2	192.168.1.15	TCP	1506	[TCP Out-Of-Order] 443 → 51093 [ACK] Seq=5889 Ack=2833 Win=64128 Len=1452
279	12.447514	87.121.121.2	192.168.1.15	TCP	1506	[TCP Out-Of-Order] 443 → 51093 [ACK] Seq=7341 Ack=2833 Win=64128 Len=1452
297	12.585423	87.121.121.2	192.168.1.15	TCP	1506	[TCP Out-Of-Order] 443 → 51093 [ACK] Seq=10245 Ack=2833 Win=64128 Len=1452
319	12.721041	87.121.121.2	192.168.1.15	TCP	1506	[TCP Out-Of-Order] 443 → 51093 [ACK] Seq=32025 Ack=2833 Win=64128 Len=1452
331	12.856911	87.121.121.2	192.168.1.15	TCP	1506	[TCP Out-Of-Order] 443 → 51093 [ACK] Seq=50901 Ack=2833 Win=64128 Len=1452

Il TCP Out-of-Order(trovato utilizzando tcp.analysis.out_of_order) ci dice che questi pacchetti indicano che i dati sono arrivati al destinatario in un ordine diverso rispetto a quello inviato dal mittente.

```
▾ [Timestamps]
    [Time since first frame in this TCP stream: 0.363204000 seconds]
    [Time since previous frame in this TCP stream: 0.048025000 seconds]
```

Questo tempo non è enorme, ma potrebbe indicare un ritardo dovuto alla congestione.

Conclusioni riguardo al progetto

Il progetto si è rivelato molto utile per comprendere i meccanismi del protocollo TCP/IP e mi ha permesso di analizzare la sequenza di handshake, il trasferimento dati HTTP e la gestione di errori come ritrasmissioni e frammentazioni. L'esempio HTTP ha evidenziato un trasferimento efficiente, mentre il download di XAAMP ha mostrato come il protocollo TCP gestisce errori e congestioni, garantendo comunque il completamento dei trasferimenti.
Ho approfondito il funzionamento del protocollo TCP/IP e ho osservato i meccanismi attraverso Wireshark. Sono soddisfatto del lavoro svolto, che mi ha dato nuove competenze utili per il futuro e la motivazione per continuare a esplorare il mondo delle reti.

Matricola: 0001098294
Nome: Gjovalin
Cognome: Mujaj