# RELAZIONE PROGETTO GESTIONE DI RETE

## **QUIC** - Quick UDP Internet Connections

Andrea Buscarinu - Michele Campus

Il progetto consiste nella implementazione di un dissector per il protocollo QUIC.
Tale protocollo a livello di trasporto, sviluppato da Google, opera sopra UDP ed è stato concepito per fornire una protezione equivalente a TLS/SSL, ridurre la latenza di comunicazione e di trasporto e stimare al meglio la larghezza di banda per evitare le congestioni di rete.

Lo sviluppo di QUIC è concentrato sulla riduzione dell' RTT per l'handshake quando si stabilisce una nuova connessione.
Il client QUIC, ad esempio, include le informazioni di sessione negoziale nel pacchetto iniziale.

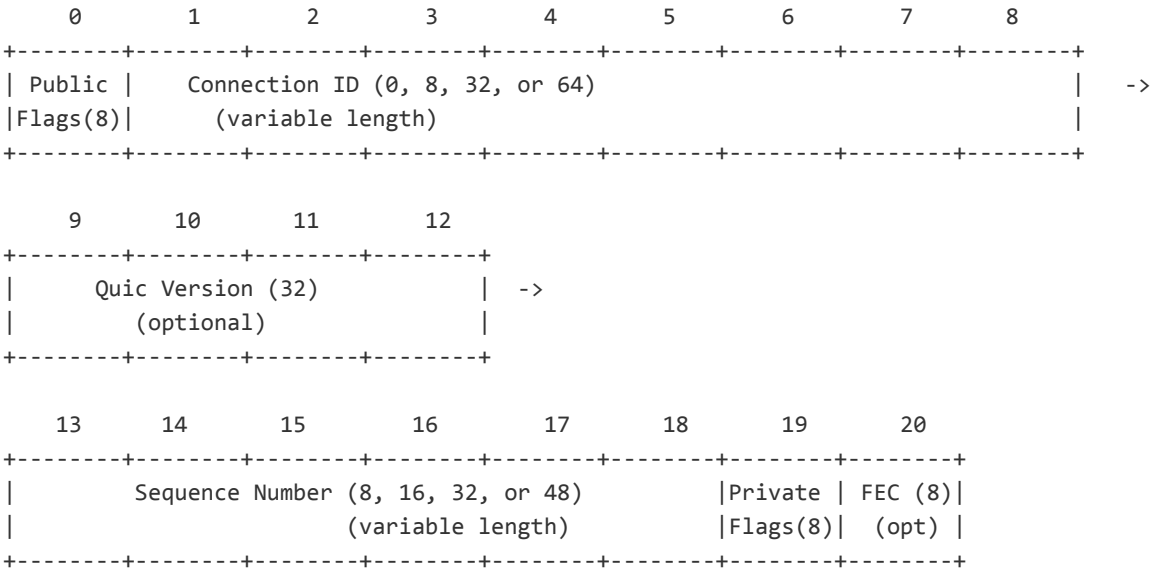
RTT TCP (Client - Server) ---> 100 ms (handshake tcp)

RTT TCP+TLS (Client - Server) ---> 100 ms (handshake tcp + handshake tls)

RTT QUIC (Client - Server) ---> 0 ms

Una volta stabilita la connessione, il client e il server si scambiano pacchetti Quic sopra UDP. Per far sì che il dissector riconosca i pacchetti Quic, si osserva l'header del pacchetto. Ogni pacchetto di questo protocollo ha un header comune con campi ben definiti. Ovviamente il livello quic si pone al di sopra di UDP (facendo riferimento alla pila ISO/OSI un pacchetto Quic farà parte della parte dati di un pacchetto UDP, quindi tale livello è al di sopra del livello trasporto).

QUIC Packet Header:



*example*:

Public Flags: 0x00

.... ...0 = Version: No
.... ..0. = Reset: No
.... 00.. = CID Length: 0 Byte (0x00)
..00 .... = Sequence Length: 1 Byte (0x00)
00.. .... = Reserved: 0x00

CID: 16707913998327591167
Sequence: 429
Payload: d8c68c86198a24005aa1e4e65aeea573a8015485be0fcb3a...

L'intestazione ha una dimensione tra 2 e 19 byte. Tutti i valori integer, compresi lunghezza, versione e tipo, sono nell'ordine dei byte little-endian.

Per prima cosa si osserva il campo **Public Flags** di 8 bit. Alcune combinazioni dei bit di tale campo danno informazioni sui campi successivi che differenziano i vari pacchetti Quic. Non tutte le permutazioni sono ammesse: questo ci dà la possibilità di scartare un pacchetto che non sia Quic.
Gli unici bit che devono rimanere a 0 sono quelli di *Reset* e *Reserved.*

**RESET**
*Bit at location 0x2, is set to indicate that the packet is a Public Reset packet.*

**RESERVED**
*Pair of bits include in 0xC0 are currently unused, and (must be) should be set to 0.*

Il campo successivo, **Connection ID**, ha una lunghezza variabile tra gli 0 e i 64 bit. Ogni pacchetto ha un proprio ID, necessario in quanto la 4-tupla IP (s_port, s_addr, d_port, d_addr) potrebbe essere insufficiente per identificare la connessione, in quanto le sessioni QUIC sono progettate per rimanere stabili, anche se il client si sposta tra reti diverse.
Nel dissector è implementata una funzione *connect_id* che calcola la lunghezza del campo CID.

**CID** - Pair of bits, included in 0xC. Within this 2 bit mask:

*0xC indicates that an 8 byte connection id is present;*
*0x8 indicates that a 4 byte connection id is present;*
*0x4 indicates that a 1 byte connection id is used;*
*0x0 indicates that the connection id is omitted.*

Il terzo campo è il **Quic Version**, anch'esso opzionale, in cui è scritta la versione del protocollo nei 4 byte occupati. Il dissector controlla ognuno dei 4 campi.

**VERSION**

*0x1 has value 1 if the packet contains a Quic Version.*

Il quarto è il **Sequence Number**, un campo di lunghezza 8, 16, 32, o 48 bits che definisce il numero di sequenza di ogni pacchetto. I numeri di sequenza
Il primo pacchetto inviato da un endpoint ha numero di sequenza uguale a **1** e ogni pacchetto successivo deve avere un numero di sequenza **n+1** rispetto a quello precedente.
Nel dissector è implementata la funzione *sequence* che calcola la lunghezza di questo campo.
I **due bit** che indicano la lunghezza del numero di sequenza sono significativi per i pacchetti di dati.

**SEQ** - Pair of bits included in 0x30. Within this 2 bit mask:

*0x30 indicates that 6 bytes of the sequence number is present;*
*0x20 indicates that 4 bytes of the sequence number is present;*
*0x10 indicates that 2 bytes of the sequence number is present;*
*0x00 indicates that 1 byte of the sequence number is present.*