**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**

# LABORATÓRIO DE REDES E SISTEMAS OPERACIONAIS

Adhonay Júnior
Daniel Leão

**Belo Horizonte**

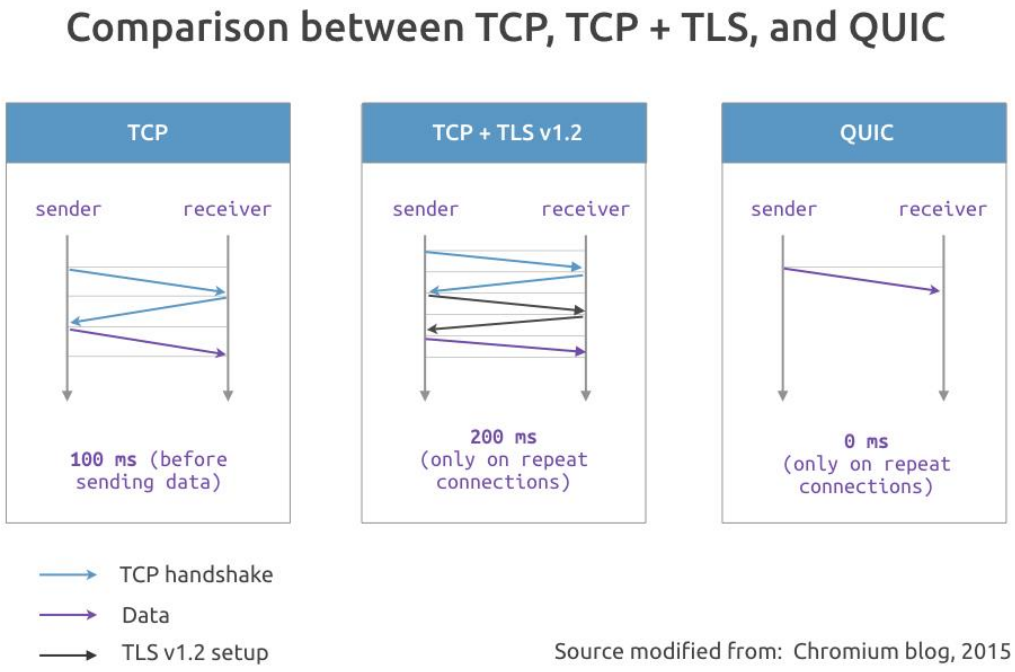
**2018**

**QUIC (Quick UDP Internet Connections)**

Na internet é comumente utilizado o protocolo TCP (que combinado com o protocolo Internet (IP), torna-se a linguagem de comunicação do núcleo da internet). UDP é significativamente mais leve do que o TCP, mas em contrapartida, dispõe de serviços de correções de erros menos eficientes que o TCP. Isto significa que o servidor de envio não está constantemente se comunicando com o servidor de recebimento para verificar se o pacote chegou e se eles chegaram na ordem correta, por exemplo. É por isso que o UDP é ótimo para serviços de jogos, pois você necessita de baixa sobrecarga reduzindo assim a latência entre o cliente e servidor. Mas para sites isso não seria ideal, pois com perda de pacotes, textos, imagens e outros itens poderiam não aparecer corretamente.

A proposta da Google é obter o melhor dos dois protocolos, mantendo a velocidade fornecida pelo UDP mas sem perder a confiabilidade e garantia da perda de pacotes. Em uma conexão TCP, um navegador demora entre dois a três ciclos até receber os pacotes de dados do servidor solicitado, com o QUIC os dados podem ser recebidos imediatamente, desde que o navegador já conheça o servidor do qual está tentando se comunicar. Plataformas como YouTube utilizam o protocolo QUIC quando estão se comunicando com o navegador Google Chrome tornando a navegação dentro das plataformas Google até 30% mais rápida. O QUIC fornece segurança e confiabilidade criptografando todos os pacotes (menos os pacotes de handshake) enviados durante a comunicação utilizando o método de troca de chaves *Diffie-Hellman* de criptografia.

O protocolo QUIC se mostra muito eficiente quando comparado com uma típica conexão TCP. Seu *handshake* é extremamente simples comparado ao TCP:



**QUIC e suas vantagens:**

- Zero roundtrip para retornar as conexões.
- Controle de Congestionamento: todos os pacotes trazem novos números de seqüência (mais fáceis de dizer quais pacotes são originais versus retransmissões), permite um cálculo preciso do tempo de ida e volta.
- Correção de erros: grupos de pacotes contêm um pacote FEC que pode ser usado para recriar um pacote perdido (sem custo de retransmissão).
- Migração de conexão: em vez de (endereço de origem, porta de origem, endereço de destino, porta de destino) é usado um ID de conexão de 64 bits (CID). Isso significa que você pode alterar o endereço IP ou as portas, mas ainda manter sua conexão viva - isso significa zero tempo de conexão mesmo se o seu celular mudar de WiFi para conexão móvel!

**Empacotamento QUIC:**

Servidor-Cliente

```
> Frame 10: 1392 bytes on wire (11136 bits), 1392 bytes captured (11136 bits) on interface 0
> Ethernet II, Src: Vmware_b7:d4:97 (00:50:56:b7:d4:97), Dst: HonHaiPr_9c:d3:7d (0c:84:dc:9c:d3:7d)
> Internet Protocol Version 4, Src: 200.150.4.209, Dst: 10.254.61.194
v User Datagram Protocol, Src Port: 443, Dst Port: 63074
    Source Port: 443
    Destination Port: 63074
    Length: 1358
    Checksum: 0x793f [unverified]
    [Checksum Status: Unverified]
    [Stream index: 1]
v QUIC (Quick UDP Internet Connections)
  > Public Flags: 0x00
    Packet Number: 64
    Payload: b176cb406326c15496856ecabb924db6a2babb53e6ab35df...
```

Cliente-Sever

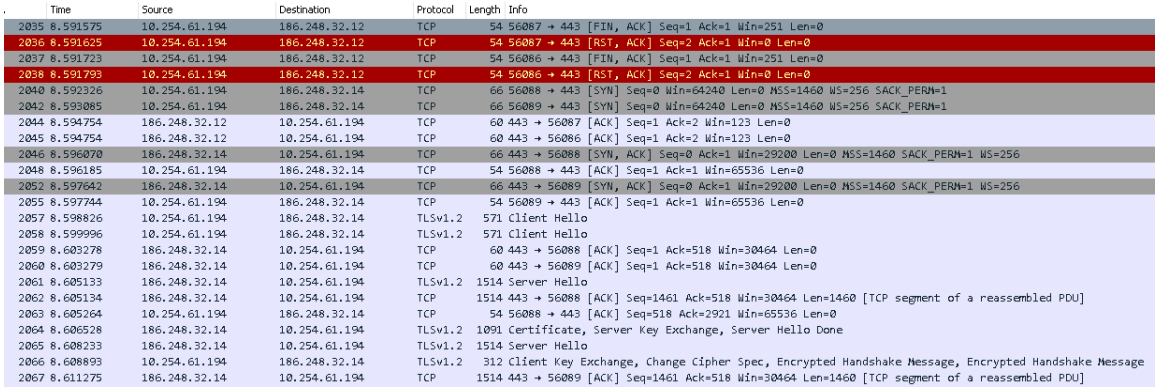
```
> Frame 11: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface 0
> Ethernet II, Src: HonHaiPr_9c:d3:7d (0c:84:dc:9c:d3:7d), Dst: Vmware_b7:d4:97 (00:50:56:b7:d4:97)
> Internet Protocol Version 4, Src: 10.254.61.194, Dst: 200.150.4.209
v User Datagram Protocol, Src Port: 63074, Dst Port: 443
    Source Port: 63074
    Destination Port: 443
    Length: 44
    Checksum: 0xdb58 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 1]
v QUIC (Quick UDP Internet Connections)
  > Public Flags: 0x0c
    CID: 2767203578736464792
    Packet Number: 246
    Payload: 45baff2abf10b441f0c90f8625a80cac59b28845ba455e74...
```

**CAPTURA DE PACOTES**

Como o tamanho do arquivo do Wireshark cresce bastante, para simplicficar foi capturado somente uma parte.Porém, apenas poucos minutos do vídeo são necessários para entender o comportamento repetitivo

e contínuo do fluxo de pacotes, quando se está assistindo um vídeo na plataforma YouTube usando o navegador Google Chrome.

O protocolo utilizado para comunicação com os servidores da plataforma Youtube Google é o QUIC. É possível notar no início da captura a troca de pacotes efetuadas para firmar o handshake entre servidor e cliente, para isto protocolos TCP e TLSv1.2 (sucessor do SSL) são trocados entre o cliente e servidor para abertura da comunicação.



| | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 2035 | 8.591575 | 10.254.61.194 | 186.248.32.12 | TCP | 54 | 56087 → 443 [FIN, ACK] Seq=1 Ack=1 Win=251 Len=0 |
| 2036 | 8.591625 | 10.254.61.194 | 186.248.32.12 | TCP | 54 | 56087 → 443 [RST, ACK] Seq=2 Ack=1 Win=0 Len=0 |
| 2037 | 8.591723 | 10.254.61.194 | 186.248.32.12 | TCP | 54 | 56086 → 443 [FIN, ACK] Seq=1 Ack=1 Win=251 Len=0 |
| 2038 | 8.591793 | 10.254.61.194 | 186.248.32.12 | TCP | 54 | 56086 → 443 [RST, ACK] Seq=2 Ack=1 Win=0 Len=0 |
| 2040 | 8.592326 | 10.254.61.194 | 186.248.32.14 | TCP | 66 | 56088 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 2042 | 8.593085 | 10.254.61.194 | 186.248.32.14 | TCP | 66 | 56089 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 2044 | 8.594754 | 186.248.32.12 | 10.254.61.194 | TCP | 60 | 443 → 56087 [ACK] Seq=1 Ack=2 Win=123 Len=0 |
| 2045 | 8.594754 | 186.248.32.12 | 10.254.61.194 | TCP | 60 | 443 → 56086 [ACK] Seq=1 Ack=2 Win=123 Len=0 |
| 2046 | 8.596070 | 186.248.32.14 | 10.254.61.194 | TCP | 66 | 443 → 56088 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=256 |
| 2048 | 8.596185 | 10.254.61.194 | 186.248.32.14 | TCP | 54 | 56088 → 443 [ACK] Seq=1 Ack=1 Win=65536 Len=0 |
| 2052 | 8.597642 | 186.248.32.14 | 10.254.61.194 | TCP | 66 | 443 → 56089 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=256 |
| 2055 | 8.597744 | 10.254.61.194 | 186.248.32.14 | TCP | 54 | 56089 → 443 [ACK] Seq=1 Ack=1 Win=65536 Len=0 |
| 2057 | 8.598826 | 10.254.61.194 | 186.248.32.14 | TLSv1.2 | 571 | Client Hello |
| 2058 | 8.599996 | 10.254.61.194 | 186.248.32.14 | TLSv1.2 | 571 | Client Hello |
| 2059 | 8.603278 | 186.248.32.14 | 10.254.61.194 | TCP | 60 | 443 → 56088 [ACK] Seq=1 Ack=518 Win=30464 Len=0 |
| 2060 | 8.603279 | 186.248.32.14 | 10.254.61.194 | TCP | 60 | 443 → 56089 [ACK] Seq=1 Ack=518 Win=30464 Len=0 |
| 2061 | 8.605133 | 186.248.32.14 | 10.254.61.194 | TLSv1.2 | 1514 | Server Hello |
| 2062 | 8.605134 | 186.248.32.14 | 10.254.61.194 | TCP | 1514 | 443 → 56088 [ACK] Seq=1461 Ack=518 Win=30464 Len=1460 [TCP segment of a reassembled PDU] |
| 2063 | 8.605264 | 10.254.61.194 | 186.248.32.14 | TCP | 54 | 56088 → 443 [ACK] Seq=518 Ack=2921 Win=65536 Len=0 |
| 2064 | 8.606528 | 186.248.32.14 | 10.254.61.194 | TLSv1.2 | 1091 | Certificate, Server Key Exchange, Server Hello Done |
| 2065 | 8.608233 | 186.248.32.14 | 10.254.61.194 | TLSv1.2 | 1514 | Server Hello |
| 2066 | 8.608893 | 10.254.61.194 | 186.248.32.14 | TLSv1.2 | 312 | Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message, Encrypted Handshake Message |
| 2067 | 8.611275 | 186.248.32.14 | 10.254.61.194 | TCP | 1514 | 443 → 56089 [ACK] Seq=1461 Ack=518 Win=30464 Len=1460 [TCP segment of a reassembled PDU] |

Notasse que existem pacotes TCP no início da comunicação, isto acontece, pois, o navegador Google Chrome dispara dois pacotes [SYN] para o servidor, afim de aumentar o desempenho, uma vez que estes pacotes serão tratados em tempos diferentes na aplicação do servidor.

Notasse também que diversos pacotes TLSv1.2 são trocados entre o navegador e servidor. O Transport Layer Security é um protocolo de segurança desenvolvido  por Tim Dierks e Eric Rescorla do IETF TLS Working Group com intuito de proteger telecomunicações de serviços de e-mail e web. O protocolo SSL provê a privacidade e a integridade de dados entre duas aplicações que comuniquem pela internet. Isso ocorre por intermédio da autenticação das partes envolvidas e da cifragem dos dados transmitidos entre as partes. Ainda, esse protocolo ajuda a prevenir que intermediários entre as duas extremidades das comunicações obtenham acesso indevido ou falsifiquem os dados que estão sendo transmitidos.

Na transferência de pacotes TLS o cliente recebe um número de identificação (CID). Este número é utilizado no protocolo QUIC para manter a sessão que fora estabelecida entre as partes servidora e cliente e toda a transferência de dados é feita através do protocolo QUIC que é diretamente empacotado sobre o UDP. Uma vez que a conexão de internet do cliente seja de alguma forma alterada, é pelo número CID que o servidor reconhecerá que aquele cliente mantivera uma conexão estabelecida em um passado arbitrário. É claro que algumas políticas referentes ao tempo em que o CID é válido são tratadas pelo servidor mas fogem deste escopo.
Após autenticação o primeiro pacote QUIC é enviado pelo cliente, um pacote do tipo Client Hello é enviado para o servidor fornecendo seu número de sessão CID e iniciando a transferência de dados. Todos os pacotes enviados do cliente para o servidor possuem número de sequência e carregam seu número de sessão. Todos os pacotes enviados do servidor para o cliente possuem número de sequência e os dados são criptografados em ambos os casos.

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 9686 | 70.568964 | 173.194.27.156 | 10.254.61.194 | QUIC | 1392 | Payload (Encrypted), PKN: 180 |
| 9687 | 70.569110 | 10.254.61.194 | 173.194.27.156 | QUIC | 84 | Payload (Encrypted), PKN: 194, CID: 13456316727904094868 |
| 9688 | 70.570983 | 173.194.27.156 | 10.254.61.194 | QUIC | 1392 | Payload (Encrypted), PKN: 181 |
| 9690 | 70.573905 | 173.194.27.156 | 10.254.61.194 | QUIC | 1392 | Payload (Encrypted), PKN: 182 |
| 9691 | 70.575326 | 173.194.27.156 | 10.254.61.194 | QUIC | 1392 | Payload (Encrypted), PKN: 183 |
| 9692 | 70.575372 | 10.254.61.194 | 173.194.27.156 | QUIC | 81 | Payload (Encrypted), PKN: 195, CID: 13456316727904094868 |
| 9693 | 70.577932 | 173.194.27.156 | 10.254.61.194 | QUIC | 1392 | Payload (Encrypted), PKN: 184 |
| 9694 | 70.579070 | 10.254.61.194 | 173.194.27.156 | QUIC | 81 | Payload (Encrypted), PKN: 196, CID: 13456316727904094868 |
| 9695 | 70.580365 | 173.194.27.156 | 10.254.61.194 | QUIC | 1392 | Payload (Encrypted), PKN: 186 |
| 9696 | 70.580637 | 10.254.61.194 | 173.194.27.156 | QUIC | 81 | Payload (Encrypted), PKN: 197, CID: 13456316727904094868 |
| 9697 | 70.581920 | 173.194.27.156 | 10.254.61.194 | QUIC | 1392 | Payload (Encrypted), PKN: 185 |
| 9698 | 70.582081 | 10.254.61.194 | 173.194.27.156 | QUIC | 78 | Payload (Encrypted), PKN: 198, CID: 13456316727904094868 |
| 9701 | 70.584319 | 173.194.27.156 | 10.254.61.194 | QUIC | 1392 | Payload (Encrypted), PKN: 187 |
| 9702 | 70.586917 | 173.194.27.156 | 10.254.61.194 | QUIC | 1392 | Payload (Encrypted), PKN: 188 |
| 9705 | 70.588755 | 173.194.27.156 | 10.254.61.194 | QUIC | 1392 | Payload (Encrypted), PKN: 189 |
| 9706 | 70.588992 | 10.254.61.194 | 173.194.27.156 | QUIC | 84 | Payload (Encrypted), PKN: 199, CID: 13456316727904094868 |
| 9707 | 70.590953 | 173.194.27.156 | 10.254.61.194 | QUIC | 1392 | Payload (Encrypted), PKN: 190 |
| 9709 | 70.593949 | 173.194.27.156 | 10.254.61.194 | QUIC | 1392 | Payload (Encrypted), PKN: 191 |
| 9710 | 70.595561 | 10.254.61.194 | 173.194.27.156 | QUIC | 81 | Payload (Encrypted), PKN: 200, CID: 13456316727904094868 |
| 9711 | 70.596160 | 173.194.27.156 | 10.254.61.194 | QUIC | 1392 | Payload (Encrypted), PKN: 192 |
| 9712 | 70.597911 | 173.194.27.156 | 10.254.61.194 | QUIC | 1392 | Payload (Encrypted), PKN: 193 |
| 9713 | 70.600585 | 173.194.27.156 | 10.254.61.194 | QUIC | 1392 | Payload (Encrypted), PKN: 194 |
| 9714 | 70.600796 | 10.254.61.194 | 173.194.27.156 | QUIC | 84 | Payload (Encrypted), PKN: 201, CID: 13456316727904094868 |

Após a solicitação de conteúdo, um padrão de comunicação pode ser observado entre cliente e servidor. O cliente envia um pacote com tamanho geralmente de 81-84 Bytes e recebe de volta dois pacotes em sequência com tamanho 1392 Bytes.