The Quic protocol in D

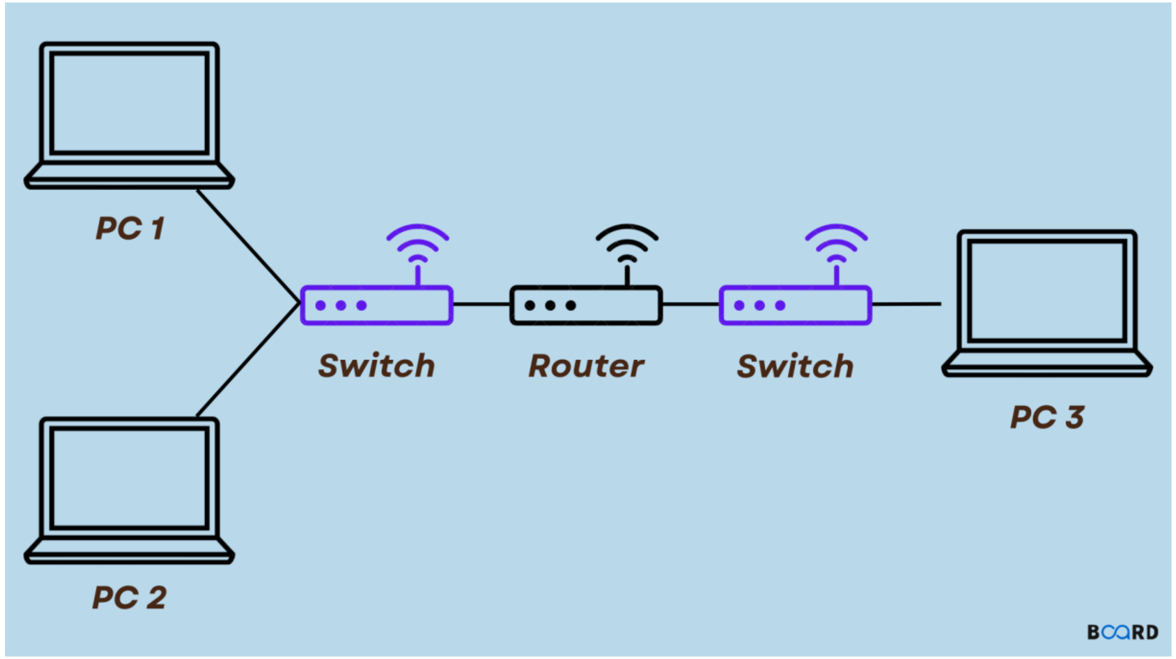
Vlad Chicoș - DCONF 2023

What is QUIC?

- Replacement for TCP
- Used by 7.4% of all the websites
- Supported by the most popular web browsers

WHY?

- Stream multiplexing, prioritization
- Does not depend on the ip stack for connection management
- Implemented in userspace, on top of UDP
  - Simpler to update
  - Middle-box tamper proof
  - Leverage the use of D



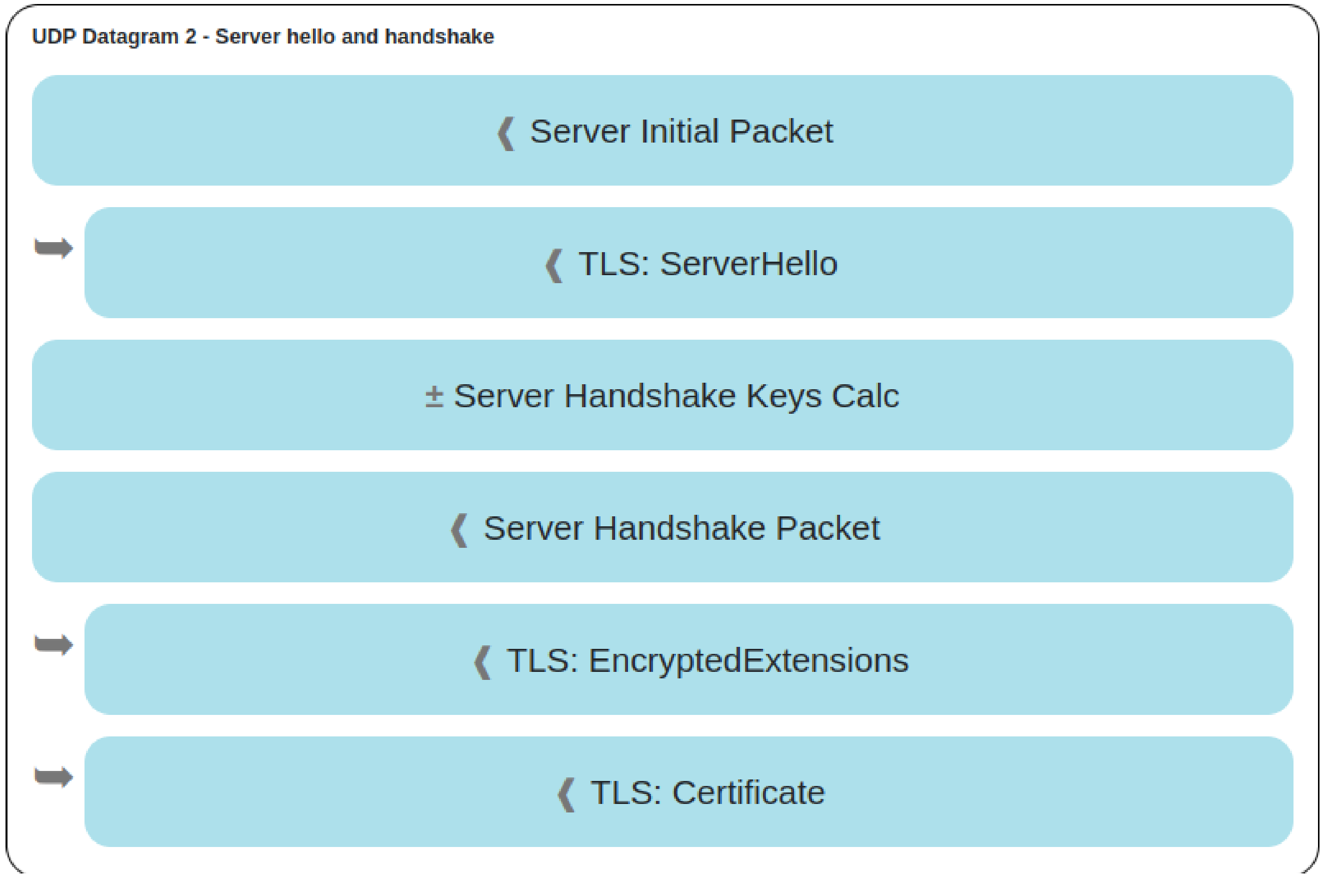
Design

- An I/O-free implementation :
    - Simpler implementation
    - Reusable
    - Easier to test
    - More modular

- Connect
- Parse datagrams
    - Moves the connection to the next state
    - Notifications
    - Adds data to the streams
- Provide datagrams
- Close

Packetization & Framing

- UDP datagrams
- QUIC packets
- QUIC frames
- TLS frames
- TLS extensions


UDP Datagram 2 - Server hello and handshake

❮ Server Initial Packet

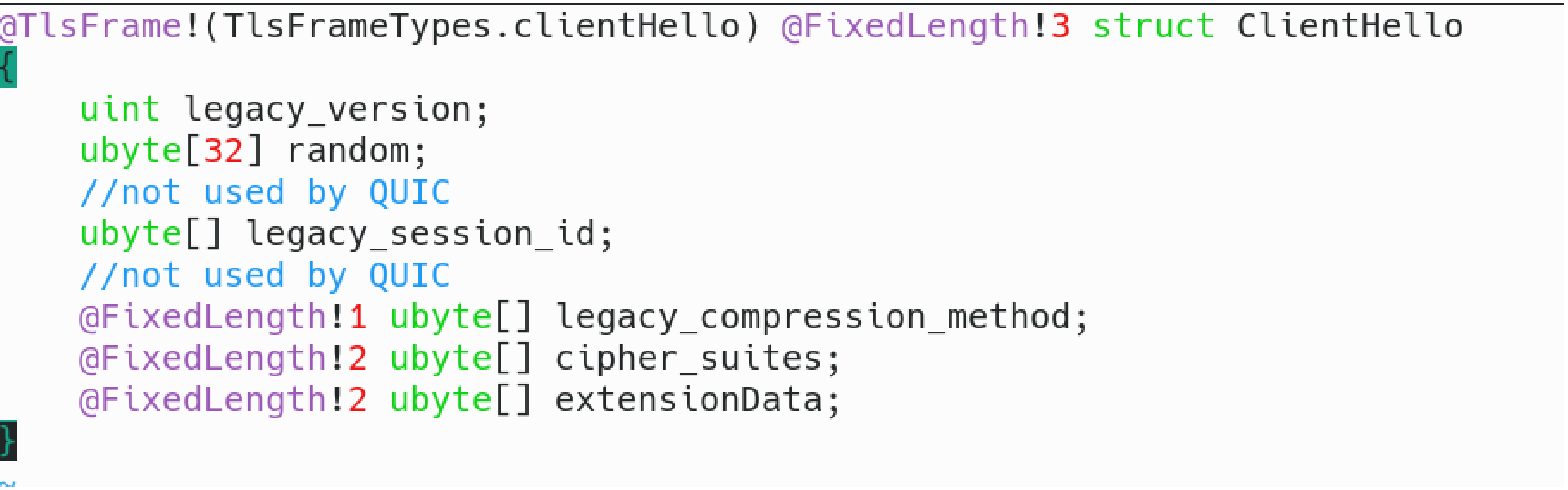
➥ ❮ TLS: ServerHello

± Server Handshake Keys Calc

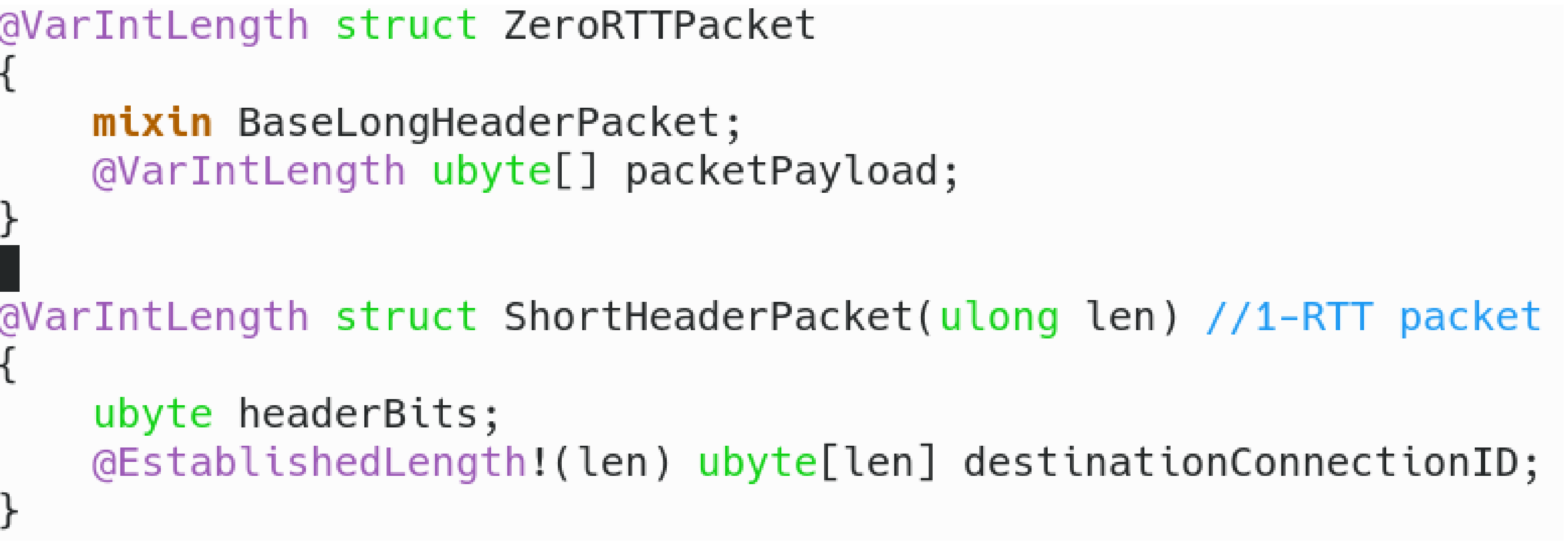
❮ Server Handshake Packet

➥ ❮ TLS: EncryptedExtensions

➥ ❮ TLS: Certificate



```
@TlsFrame!(TlsFrameTypes.clientHello) @FixedLength!3 struct ClientHello
{
    uint legacy_version;
    ubyte[32] random;
    //not used by QUIC
    ubyte[] legacy_session_id;
    //not used by QUIC
    @FixedLength!1 ubyte[] legacy_compression_method;
    @FixedLength!2 ubyte[] cipher_suites;
    @FixedLength!2 ubyte[] extensionData;
}
```



```
@VarIntLength struct ZeroRTTPacket
{
    mixin BaseLongHeaderPacket;
    @VarIntLength ubyte[] packetPayload;
}

@VarIntLength struct ShortHeaderPacket(ulong len) //1-RTT packet
{
    ubyte headerBits;
    @EstablishedLength!(len) ubyte[len] destinationConnectionID;
}
```

```
....................
  auto opDispatch(string name)() {
      alias FieldType = typeof(__traits(getMember, FrameType, name));

      alias attributes = __traits(getAttributes, mixin(FrameType.stringof, '.', name));
      ....................
      FieldType value;

      else static if (is(FieldType == AckRange))
      {
          static assert(attributes.length == 0, errorPrefix!() ~ ": no attributes allowed");
          decodeVarInt(value.gap, networkStream, bufIndex);
          decodeVarInt(value.rangeLength, networkStream, bufIndex);
          return value;
      }

      else static if (is(FieldType == ubyte[]))
      {
          static assert(attributes.length == 1, errorPrefix
            ~ ": must have exactly one attribute @VarIntLength or @FixedLength!n");

          static if (is(attributes[0] == VarIntLength))
          {
              VarInt len;
              decodeVarInt(len, networkStream, bufIndex);
              bufIndex += len;
              return networkStream[bufIndex-len..bufIndex];
          }

          static if (hasFixedLength!(attributes[0]))
          {
              auto lenOfLength = getFixedLength!(attributes[0]);
              auto len = readBigEndianField(networkStream, bufIndex, lenOfLength);
              bufIndex += len;
              return networkStream[bufIndex-len..bufIndex];
          }
      }
```

```
void getBytes(Writer, Frame)(ref Writer writer, Frame F)
    {
        LocalAppender wLocal;
        foreach(i, field; F.tupleof)
        {
            alias attributes = __traits(getAttributes, F.tupleof[i]);

            static if (is(typeof(field) == VarInt))
            {
                encodeVarInt(wLocal, field);
            }
...........................

            else static if (attributes.length > 0)
            {

                static if(is(attributes == VarIntLength))
                {
                    encodeVarInt(wLocal, field.length);
                    wLocal ~= field;
                }
...........................
```

Crypto.d - openssl bindings

- Header protection
- Key generation
- Certificate verification
- Create some interfaces

Quic streams – bidirectional streams:

- More testing

Thank you!