Joris Herbots
Ph.D. Student @ syncedSheep
joris.herbots@uhasselt.be
Quick introduction to QUIC

- **QUIC** is a name, not an acronym
- A **general-purpose** transport layer protocol
- Standardized by the **IETF** in **May 2021**
- You have probably been using it (a lot)
  - Firefox and Chromium-based browsers
  - All Google applications, Facebook, Instagram, Youtube, Apple OS …
Call it Quick UDP Internet Connections one more time

I dare you
Quick introduction to QUIC

- **Encryption** by default
  - Main driver **against ossification**
  - Great for preventing third parties from interfering with our information
  - Less great for research and development
Quick introduction to QUIC

- **Encryption** by default
  - Main driver against ossification
  - Great for preventing third parties from interfering with our information
  - Less great for research and development

- **Implemented on UDP in user space**
  - Lowers the threshold for experimenting with the protocol!
  - At present, 25 implementations* exist in a plethora of languages

---

Quick introduction to QUIC

● **Encryption** by default
  ○ Main driver against ossification
  ○ Great for preventing third parties from interfering with our information
  ○ Less great for research and development

● **Implemented on UDP in user space**
  ○ Lowers the threshold for experimenting with the protocol!
  ○ At present, 25 implementations* exist in a plethora of languages

● **HTTP/3**

---

Let’s start experimenting with QUIC-HTTP/3!
Let’s start simple and connect to an existing HTTP/3 server… Simple, right?

This site can’t be reached

quic.aiortc.org refused to connect.

Try:
- Checking the connection
- Checking the proxy and the firewall

ERR_CONNECTION_REFUSED
A tale of browsers, HTTP/3 and the alt-svc header

Firefox and Chrome utilize the `alt-svc` HTTP header to discover HTTP/3 servers.

So how can we test HTTP/3-only servers?

Chrome allows overriding this with `--origin-to-force-quic-on`.

Firefox requires a new `about:config` entry `network.http.http3.alt-svc-mapping-for-testing`
Now let’s look at what happened under the hood

- Remember, QUIC is by default encrypted

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>436</td>
<td>3.429118</td>
<td>193.167.100.100</td>
<td>193.167.100.100</td>
<td>QUIC</td>
<td>69</td>
<td>Protected Payload (KP0)</td>
</tr>
<tr>
<td>437</td>
<td>3.429379</td>
<td>193.167.100.100</td>
<td>193.167.100.100</td>
<td>QUIC</td>
<td>76</td>
<td>Protected Payload (KP0), DCID=b95d3eff</td>
</tr>
<tr>
<td>438</td>
<td>3.429460</td>
<td>193.167.100.100</td>
<td>193.167.100.100</td>
<td>QUIC</td>
<td>1282</td>
<td>Protected Payload (KP0)</td>
</tr>
<tr>
<td>439</td>
<td>3.429801</td>
<td>193.167.100.100</td>
<td>193.167.100.100</td>
<td>QUIC</td>
<td>1282</td>
<td>Protected Payload (KP0)</td>
</tr>
<tr>
<td>440</td>
<td>3.430143</td>
<td>193.167.100.100</td>
<td>193.167.100.100</td>
<td>QUIC</td>
<td>1282</td>
<td>Protected Payload (KP0)</td>
</tr>
<tr>
<td>441</td>
<td>3.430485</td>
<td>193.167.100.100</td>
<td>193.167.100.100</td>
<td>QUIC</td>
<td>1282</td>
<td>Protected Payload (KP0)</td>
</tr>
<tr>
<td>442</td>
<td>3.430735</td>
<td>193.167.100.100</td>
<td>193.167.100.100</td>
<td>QUIC</td>
<td>1282</td>
<td>Protected Payload (KP0)</td>
</tr>
<tr>
<td>443</td>
<td>3.430827</td>
<td>193.167.100.100</td>
<td>193.167.100.100</td>
<td>QUIC</td>
<td>1282</td>
<td>Protected Payload (KP0)</td>
</tr>
<tr>
<td>444</td>
<td>3.431169</td>
<td>193.167.100.100</td>
<td>193.167.100.100</td>
<td>QUIC</td>
<td>1282</td>
<td>Protected Payload (KP0)</td>
</tr>
</tbody>
</table>

- Frame 444: 1282 bytes on wire (10256 bits), 1282 bytes captured
- Protocol: Point-to-Point Protocol
- Internet Protocol Version 4, Src: 193.167.100.100, Id: 443
- User Datagram Protocol, Src Port: 443, Dst Port: 554
- QUIC IETF

Encrypted payload
Now let’s look at what happened under the hood

- Remember, QUIC is by default encrypted
- Most TLS backends support the `SSLKEYLOGFILE` environment variable

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>436</td>
<td>3.429118</td>
<td>193.167.100.100</td>
<td>193.167.0.100</td>
<td>QUIC</td>
<td>69 Protected Payload (KP0), DCID=3a72c3f9f03ad</td>
</tr>
<tr>
<td>437</td>
<td>3.429370</td>
<td>193.167.0.100</td>
<td>193.167.100.100</td>
<td>QUIC</td>
<td>76 Protected Payload (KP0), DCID=b95d3eff, PKN:</td>
</tr>
<tr>
<td>438</td>
<td>3.429460</td>
<td>193.167.0.100</td>
<td>193.167.0.100</td>
<td>HTTP/3</td>
<td>1282 Protected Payload (KP0), DCID=3a72c3f9f03ad</td>
</tr>
<tr>
<td>439</td>
<td>3.429601</td>
<td>193.167.0.100</td>
<td>193.167.0.100</td>
<td>HTTP/3</td>
<td>1282 Protected Payload (KP0), DCID=3a72c3f9f03ad</td>
</tr>
<tr>
<td>440</td>
<td>3.430143</td>
<td>193.167.0.100</td>
<td>193.167.0.100</td>
<td>HTTP/3</td>
<td>1282 Protected Payload (KP0), DCID=3a72c3f9f03ad</td>
</tr>
<tr>
<td>441</td>
<td>3.430485</td>
<td>193.167.0.100</td>
<td>193.167.0.100</td>
<td>HTTP/3</td>
<td>1282 Protected Payload (KP0), DCID=3a72c3f9f03ad</td>
</tr>
<tr>
<td>442</td>
<td>3.430735</td>
<td>193.167.0.100</td>
<td>193.167.0.100</td>
<td>HTTP/3</td>
<td>1282 Protected Payload (KP0), DCID=3a72c3f9f03ad</td>
</tr>
<tr>
<td>443</td>
<td>3.430827</td>
<td>193.167.0.100</td>
<td>193.167.0.100</td>
<td>HTTP/3</td>
<td>1282 Protected Payload (KP0), DCID=3a72c3f9f03ad</td>
</tr>
</tbody>
</table>

Frame 444: 1282 bytes on wire (18256 bits), 1282 bytes captured (182-)

- Point-to-Point Protocol
- Internet Protocol Version 4, Src: 193.167.100.100, Dst: 193.167.0.100
- User Datagram Protocol, Src Port: 443, Dst Port: 55094
- QUIC IETF
  - QUIC Connection information [Packet Length: 1252]
  - QUIC Short Header DCID=3a72c3f9f03ad9e PKN=364
  - STREAM id=0 fin=0 off=31617/5 len=1219 dir=Bidirectional origin=Client

Decrypted payload
Now let’s look at what happened under the hood

- Structured endpoint logging with [qlog](https://github.com/JorisHerbots/vegvisir)
- `<qvis>` visualization tools for QUIC and HTTP/3
- Great FOSDEM talk about qlog and qvis:
  https://archive.fosdem.org/2021/schedule/event/webperf_quic_http3_qlog_qvis/
Now let’s look at what happened under the hood

- **<qvis> tool**: [https://qvis.quictools.info/](https://qvis.quictools.info/)
Let’s set up our own QUIC server and client

- Each implementation has its own installation and requirements
- Different implementations have different performance characteristics
  - E.g., More tuned towards a certain scenario, support for feature X and Y, …
- Requires setting up (self-signed) certificates for encryption
- Some have weird quirks

```go
67 // NewCubicSender makes a new cubic sender
68 func NewCubicSender(
```
Let’s set up our own QUIC server and client

- Each implementation has its own installation and requirements
- Different implementations have different performance characteristics
  - E.g., More tuned towards a certain scenario, support for feature X and Y, ...
- Requires setting up (self-signed) certificates for encryption
- Some have weird quirks

```go
// NewCubicSender makes a new cubic sender
func NewCubicSender(
    clock Clock,
    rttStats *utils.RTStats,
    initialMaxDatagramSize protocol.ByteCount,
    reno bool,
    tracer logging.ConnectionTracer,
) *cubicSender {
⇒ Testing them all – or even a hand-picked selection – takes time
```
A single test setup

SERVER

Docker Container

SHAPER

SCENARIO

OR

CLIENT

Docker Container

CLI Command

CLIENT
Steering Vegvisir with JSON configurations

📦 Implementation Configuration
  - Defines **what** entities are available
  - Declares **parameters**

⚙️ Experiment Configuration
  - Defines **how** to use the picked entities
  - Provides **arguments**
  - Configures **sensors**
  - \#tests = \#servers × \#clients × \#shapers

Loose coupling
Implementation configuration example

```json
{
    "clients": {
        "aiocquic": {
            "image": "aiortc/aiocquic-qns",
            "parameters": {"REQUESTS": true}
        },
        "quicly": {...}
    },
    "servers": {
        "quic-go": {
            "image": "martenseemann/quic-go-interop:latest"
        },
        "mwfst": {...}
    },
    "shapers": {
        "tc-netem": {
            "image": "tc-netem",
            "scenarios": {
                "simple": {
                    "command": "\"simple ![LATENCY] ![THROUGHPUT]\"",
                    "parameters": ["THROUGHPUT", "LATENCY"]
                },
                "cellular-loss-good": "\"akamai_cellular_emulation.sh loss_based good\"
            }
        }
    }
}
```
Implementation configuration example

```json
{
    "clients": {
        "aiogic": {
            "image": "aiortc/aiogic-qns",
            "parameters": {"REQUESTS": true}
        },
        "quicly": {...}
    },
    "servers": {
        "quic-go": {
            "image": "martenseemann/quic-go-interop:latest"
        },
        "mvfst": {...}
    },
    "shapers": {
        "tc-netem": {
            "image": "tc-netem",
            "scenarios": {
                "simple": {
                    "command": "\"simple !{LATENCY} !{THROUGHPUT}\"",
                    "parameters": ["THROUGHPUT", "LATENCY"]
                },
                "cellular-loss-good": "\"akamai_cellular_emulation.sh loss_based good\"
            }
        }
    }
}
```
Implementation configuration CLI client example

```json
{
  "clients": {
    "chrome": {
      "parameters": {
        "REQUEST_URL": true
      },
      "command": "google-chrome-stable --origin-to-force-quic-on={ORIGIN} -p{ORIGIN_PORT} --enable-experimental-web-platform-features --log-net-log={LOG_PATH_CLIENT}/net-log.json --autoplay-policy=no-user-gesture-required --auto-open-devtools-for-tabs --ignore-certificate-errors-spki-list={CERT_FINGERPRINT} -r{REQUEST_URL}",
      "construct": [
        {
          "root_required": false,
          "command": "python ./util/chrome-set-downloads-folder.py -/.config/google-chrome/Default/Preferences "-{DOWNLOAD_PATH_CLIENT}\"
        }
      ]
    }
  }
}
```

System parameters

CLI command setup
Experiment configuration example

```json
{
    "clients": [
        {
            "name": "aioguc",
            "arguments": {
                "REQUESTS": "https://!\{ORIGIN\}/1MB.bin"
            }
        },
        {
            "name": "tc-netem",
            "log_name": "tc-netem-cellular-experience-good",
            "scenario": "cellular-experience-good"
        },
        {
            "name": "ns3-quick",
            "scenario": "simple-p2p",
            "arguments": {
                "THROUGHPUT": "30",
                "LATENCY": "10"
            }
        }
    ],
    "servers": [
        {
            "name": "aioguc"
        },
        {
            "name": "quic-go"
        },
        {
            "name": "ngtcp2"
        }
    ]
}
```
Experiment configuration example

```json
1 {
2   "clients": [
3     {
4       "name": "aioguc",
5       "arguments": {
6         "REQUESTS": "https://!{ORIGIN}/1MB.bin"
7       }
8     },
9   ],
10  "shapers": [
11    {
12      "name": "tc-netem",
13      "log_name": "tc-netem-cellular-experience-good",
14      "scenario": "cellular-experience-good"
15    },
16    {
17      "name": "ns3-quic",
18      "scenario": "simple-p2p",
19      "arguments": {
20        "THROUGHPUT": "30",
21        "LATENCY": "10"
22      }
23    }
24  ],
25  "servers": [
26    {"name": "aioguc"},
27    {"name": "quic-go"},
28    {"name": "ngtcp2"}
29  ]
30 }
```

Shaper scenarios
Experiment configuration example

```json
{
    "clients": [
        {
            "name": "aioguc",
            "arguments": {
                "REQUESTS": "https://!{ORIGIN}/1MB.bin"
            }
        }
    ],
    "shapers": [
        {
            "name": "tc-netem",
            "log_name": "tc-netem-cellular-experience-good",
            "scenario": "cellular-experience-good"
        },
        {
            "name": "ns3-quick",
            "scenario": "simple-p2p",
            "arguments": {
                "THROUGHPUT": "30",
                "LATENCY": "10"
            }
        }
    ],
    "servers": [
        {"name": "aioguc"},
        {"name": "quic-go"},
        {"name": "ngtcp2"}
    ]
}
```

1 client

×

2 shapers

×

3 servers

6 tests
Experiment configuration example

```
"environment": {
  "name": "webserver-basic",
  "sensors": [
  {
    "name": "timeout",
    "timeout": 30
  }
  ],
},

"settings": {
  "label": "fosdem_example",
  "www_dir": "./www",
  "iterations": 1
}
```

Sensor config
Experiment output

- 2023-01-27T_16-49-28
  - 30 items
  - 27 Jan

- 2023-02-03T_12-48-38
  - 30 items
  - Yesterday

- aloquic_ns3-quic_aioquic
  - 5 items
  - Yesterday

- aloquic_ns3-quic_ngtcp2
  - 5 items
  - Yesterday

- aloquic_ns3-quic_quic-go
  - 5 items
  - Yesterday

- aloquic_tc-netem-cellular-experience-good_aioquic
  - 5 items
  - Yesterday

- aloquic_tc-netem-cellular-experience-good_ngtcp2
  - 5 items
  - Yesterday

- aloquic_tc-netem-cellular-experience-good_quic-go
  - 5 items
  - Yesterday

- aloquic_tc-netem-cellular-loss-median_aioquic
  - 5 items
  - Yesterday
Experiment output

Output folders

- Auto-generated by Vegvisir and mounted to the Docker containers
- Each docker container can write “anything” to `/logs`
- Client additionally has a `/downloads` folder
Extensibility

Sensors

- Currently available:
  - timeout sensor
  - browser download sensor
- Ability to create custom sensors for your experiments by extending abstract base class `ABCSensor`

Hooks

- Broad applicability means having little knowledge about experiments
- Program custom behavior with `pre_run_hook` and `post_run_hook` by extending the `BaseEnvironment` class

https://github.com/JorisHerbots/vegvisir/tree/master/vegvisir/environments
Thank you!