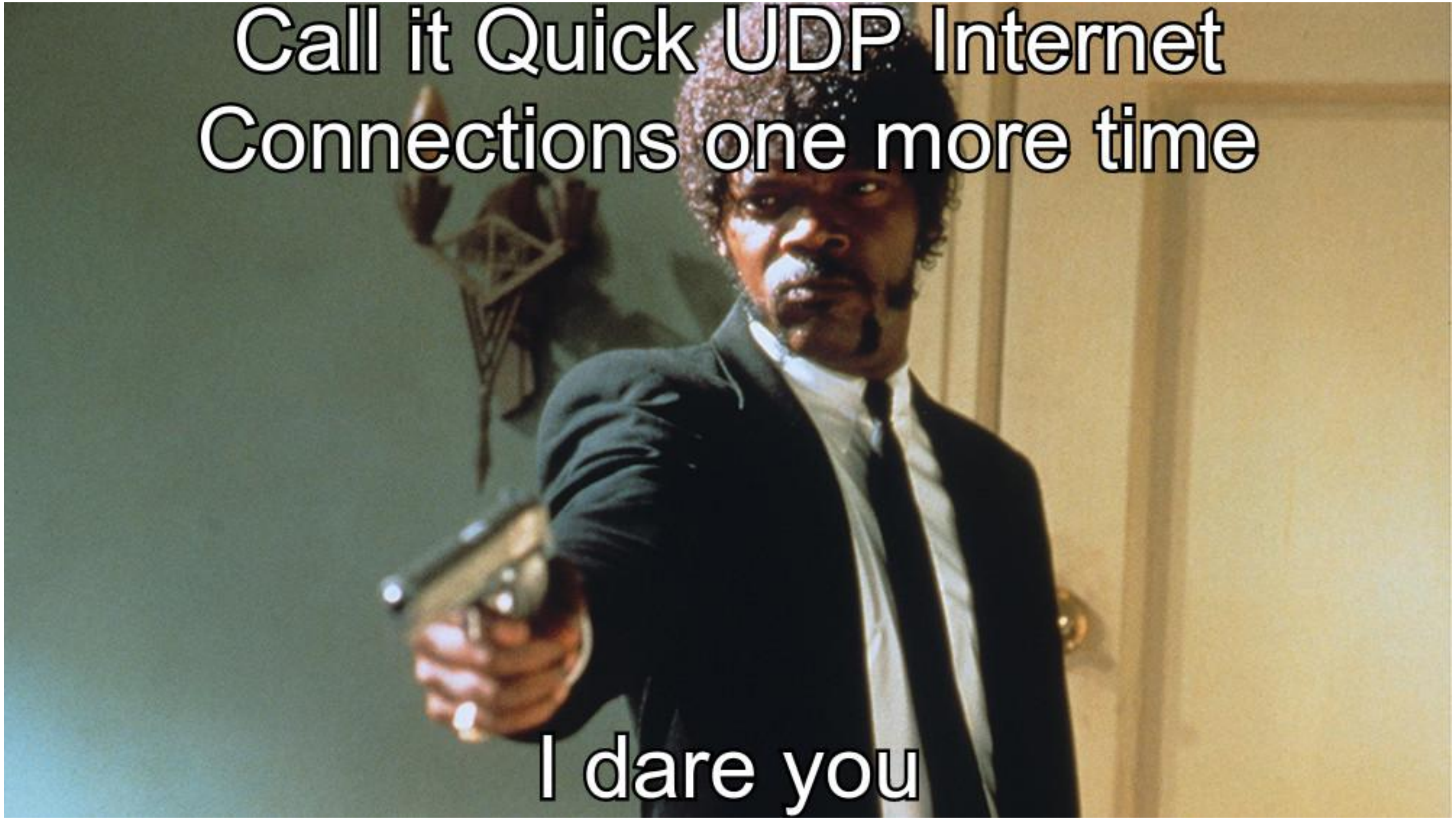
VEGVISIR

Joris Herbots
Ph.D. Student @ ▸▸ UHASSELT EDM

@syncedSheep
joris.herbots@uhasselt.be

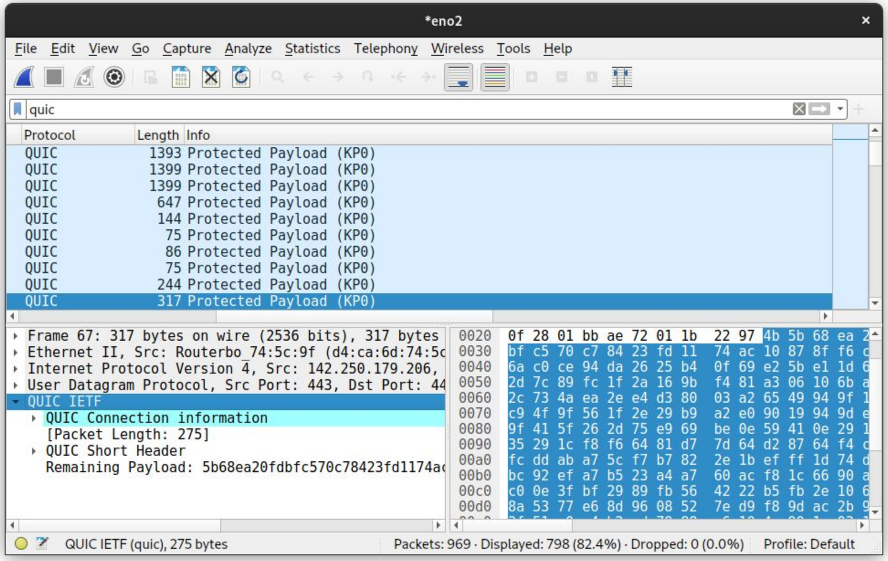
Quick introduction to QUIC

- **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


*eno2
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
quic

```
Protocol        Length Info
QUIC              1393 Protected Payload (KP0)
QUIC              1399 Protected Payload (KP0)
QUIC              1399 Protected Payload (KP0)
QUIC               647 Protected Payload (KP0)
QUIC               144 Protected Payload (KP0)
QUIC                75 Protected Payload (KP0)
QUIC                86 Protected Payload (KP0)
QUIC                75 Protected Payload (KP0)
QUIC               244 Protected Payload (KP0)
QUIC               317 Protected Payload (KP0)
```

```
Frame 67: 317 bytes on wire (2536 bits), 317 bytes
Ethernet II, Src: Routerbo_74:5c:9f (d4:ca:6d:74:5c
Internet Protocol Version 4, Src: 142.250.179.206,
User Datagram Protocol, Src Port: 443, Dst Port: 44
QUIC IETF
  QUIC Connection information
  [Packet Length: 275]
  QUIC Short Header
  Remaining Payload: 5b68ea20fdbfc570c78423fd1174ac
```

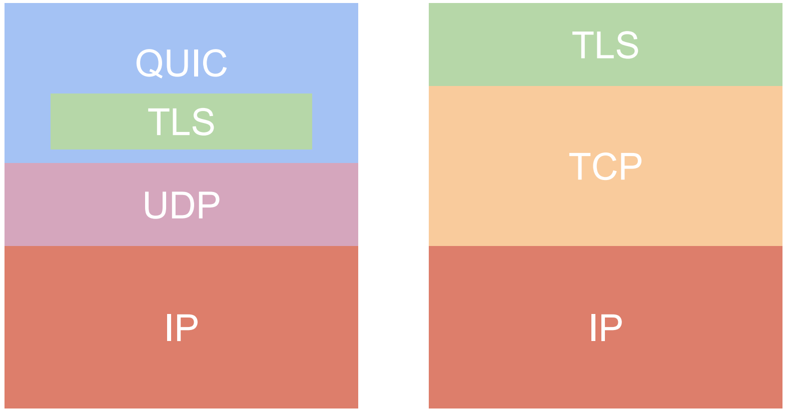
```
0020  0f 28 01 bb ae 72 01 1b  22 97 4b 5b 68 ea 2
0030  bf c5 70 c7 84 23 fd 11  74 ac 10 87 8f f6 c
0040  6a c0 ce 94 da 26 25 b4  0f 69 e2 5b e1 1d 6
0050  2d 7c 89 fc 1f 2a 16 9b  f4 81 a3 06 10 6b a
0060  2c 73 4a ea 2e e4 d3 80  03 a2 65 49 94 9f 1
0070  c9 4f 9f 56 1f 2e 29 b9  a2 e0 90 19 94 9d e
0080  9f 41 5f 26 2d 75 e9 69  be 0e 59 41 0e 29 1
0090  35 29 1c f8 f6 64 81 d7  7d 64 d2 87 64 f4 c
00a0  fc dd ab a7 5c f7 b7 82  2e 1b ef ff 1d 74 d
00b0  bc 92 ef a7 b5 23 a4 a7  60 ac f8 1c 66 90 a
00c0  c0 0e 3f bf 29 89 fb 56  42 22 b5 fb 2e 10 6
00d0  8a 53 77 e6 8d 96 08 52  7e d9 f8 9d ac 2b 9
```

QUIC IETF (quic), 275 bytes Packets: 969 · Displayed: 798 (82.4%) · Dropped: 0 (0.0%) Profile: Default

https://github.com/JorisHerbots/vegvisir

Quick introduction to QUIC

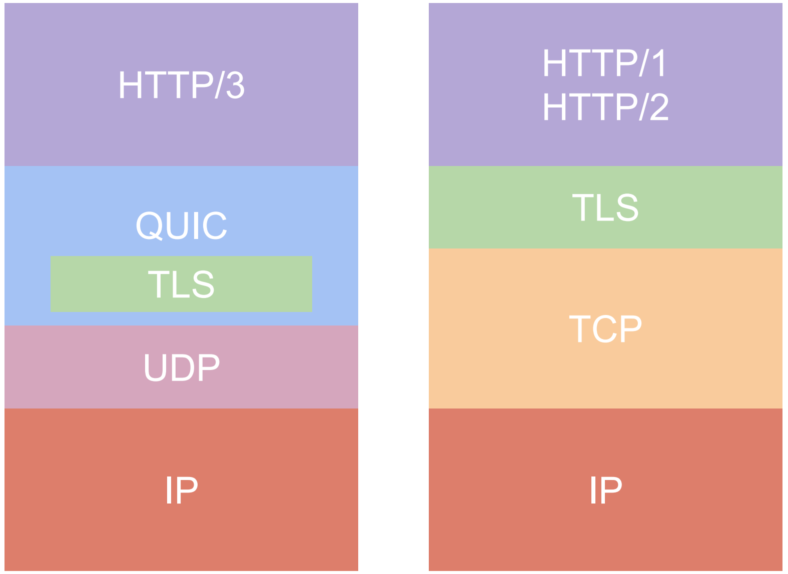
- **Encryption** by default
  - Main driver **against ossification**
  - Great for preventing third parties from interfering with our information
  - Less great for research and development
- Implemented on **UDP** in **user space**
  - Lowers the threshold for experimenting with the protocol!
  - At present, **25** implementations* exist in a plethora of languages



* https://github.com/quicwg/base-drafts/wiki/Implementations

Quick introduction to QUIC

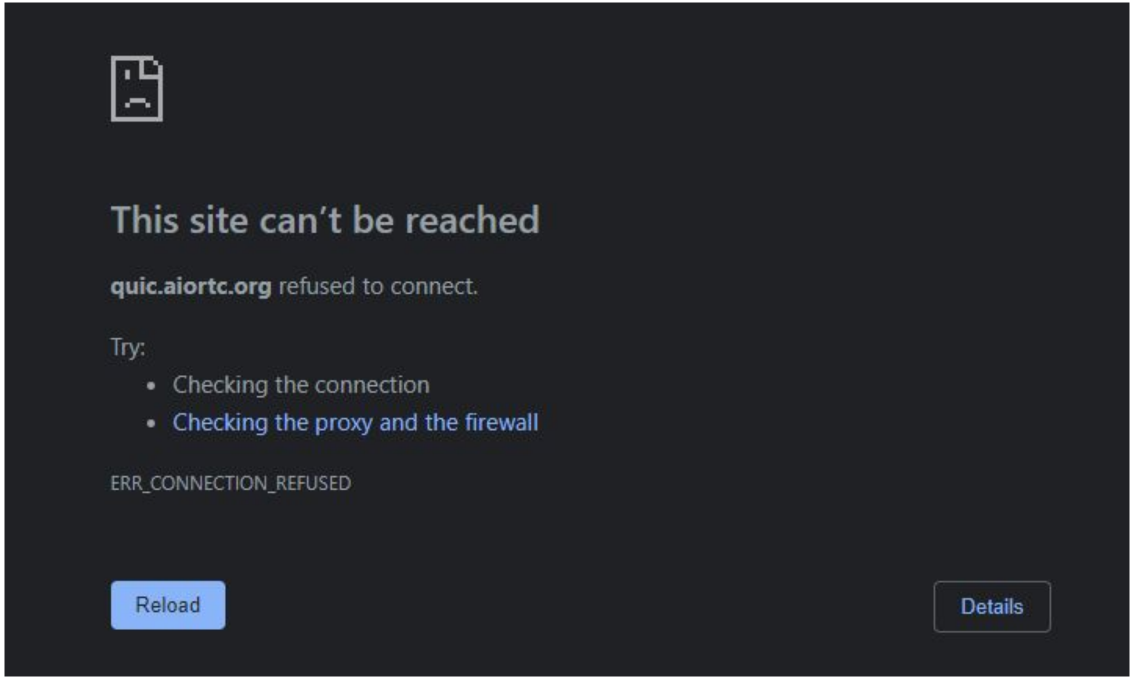
- **Encryption** by default
  - Main driver **against ossification**
  - Great for preventing third parties from interfering with our information
  - Less great for research and development
- Implemented on **UDP** in **user space**
  - Lowers the threshold for experimenting with the protocol!
  - At present, **25** implementations* exist in a plethora of languages
- **HTTP/3**



* https://github.com/quicwg/base-drafts/wiki/Implementations

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

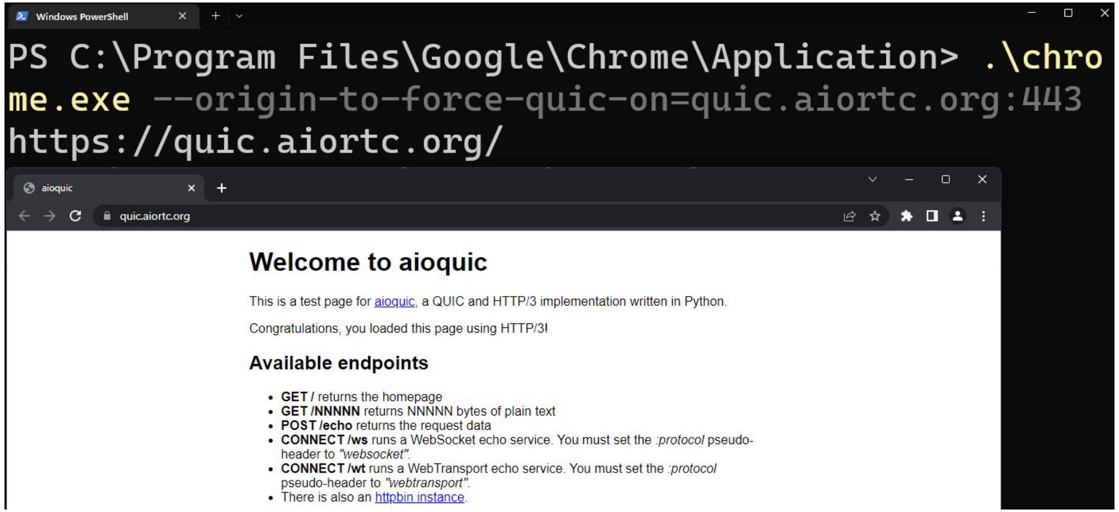
A tale of browsers, HTTP/3 and the alt-svc header

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`



```
PS C:\Program Files\Google\Chrome\Application> .\chrome.exe --origin-to-force-quic-on=quic.aiortc.org:443 https://quic.aiortc.org/
```

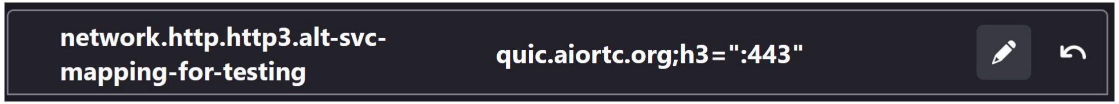
Welcome to aioquic

This is a test page for aioquic, a QUIC and HTTP/3 implementation written in Python.

Congratulations, you loaded this page using HTTP/3!

Available endpoints

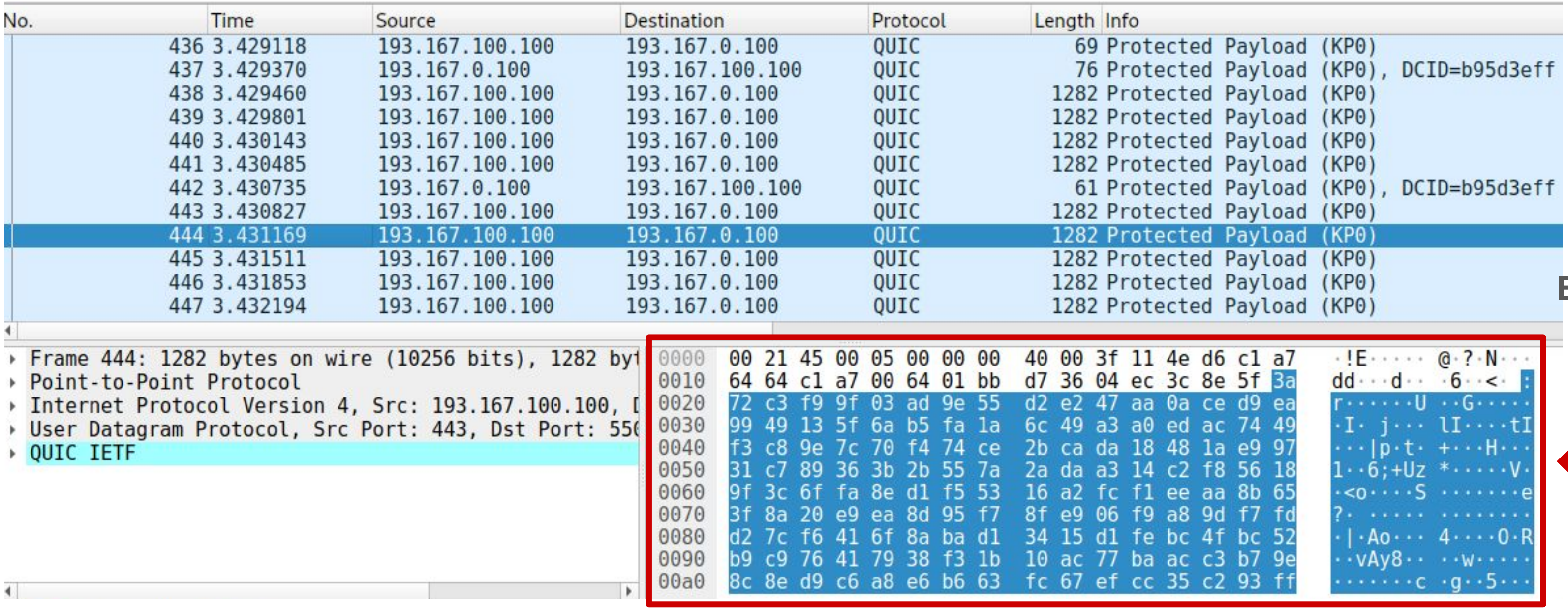
- **GET /** returns the homepage
- **GET /NNNNN** returns NNNNN bytes of plain text
- **POST /echo** returns the request data
- **CONNECT /ws** runs a WebSocket echo service. You must set the *:protocol* pseudo-header to *"websocket"*.
- **CONNECT /wt** runs a WebTransport echo service. You must set the *:protocol* pseudo-header to *"webtransport"*.
- There is also an httpbin instance.



network.http.http3.alt-svc-mapping-for-testing	**quic.aiortc.org;h3=":443"**

Now let's look at what happened under the hood

- Remember, QUIC is by default encrypted



```
No.   Time      Source           Destination      Protocol  Length Info
  436 3.429118  193.167.100.100  193.167.0.100    QUIC        69 Protected Payload (KP0)
  437 3.429370  193.167.0.100    193.167.100.100  QUIC        76 Protected Payload (KP0), DCID=b95d3eff
  438 3.429460  193.167.100.100  193.167.0.100    QUIC      1282 Protected Payload (KP0)
  439 3.429801  193.167.100.100  193.167.0.100    QUIC      1282 Protected Payload (KP0)
  440 3.430143  193.167.100.100  193.167.0.100    QUIC      1282 Protected Payload (KP0)
  441 3.430485  193.167.100.100  193.167.0.100    QUIC      1282 Protected Payload (KP0)
  442 3.430735  193.167.0.100    193.167.100.100  QUIC        61 Protected Payload (KP0), DCID=b95d3eff
  443 3.430827  193.167.100.100  193.167.0.100    QUIC      1282 Protected Payload (KP0)
  444 3.431169  193.167.100.100  193.167.0.100    QUIC      1282 Protected Payload (KP0)
  445 3.431511  193.167.100.100  193.167.0.100    QUIC      1282 Protected Payload (KP0)
  446 3.431853  193.167.100.100  193.167.0.100    QUIC      1282 Protected Payload (KP0)
  447 3.432194  193.167.100.100  193.167.0.100    QUIC      1282 Protected Payload (KP0)
```

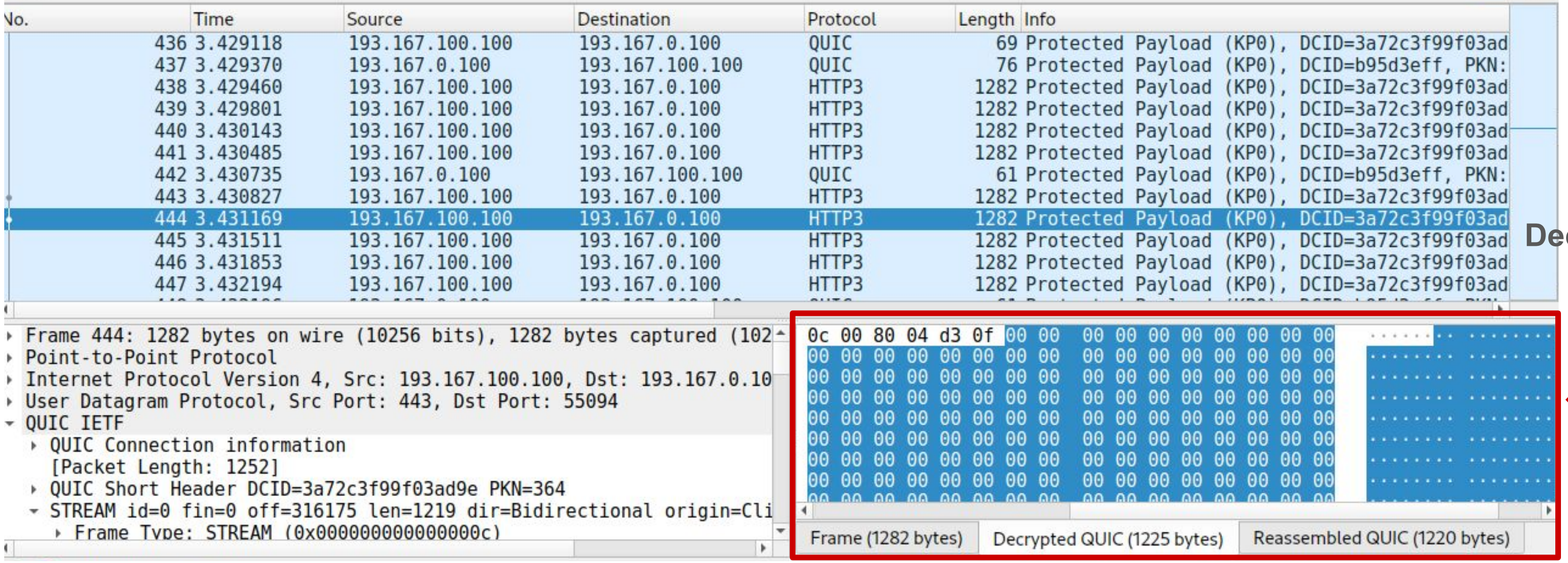
```
▸ Frame 444: 1282 bytes on wire (10256 bits), 1282 byt
▸ Point-to-Point Protocol
▸ Internet Protocol Version 4, Src: 193.167.100.100, D
▸ User Datagram Protocol, Src Port: 443, Dst Port: 55
▸ QUIC IETF
```

```
0000  00 21 45 00 05 00 00 00  40 00 3f 11 4e d6 c1 a7   ·!E····· @·?·N···
0010  64 64 c1 a7 00 64 01 bb  d7 36 04 ec 3c 8e 5f 3a   dd···d·· ·6··<·_:
0020  72 c3 f9 9f 03 ad 9e 55  d2 e2 47 aa 0a ce d9 ea   r······U ··G·····
0030  99 49 13 5f 6a b5 fa 1a  6c 49 a3 a0 ed ac 74 49   ·I·_j··· lI····tI
0040  f3 c8 9e 7c 70 f4 74 ce  2b ca da 18 48 1a e9 97   ···|p·t· +···H···
0050  31 c7 89 36 3b 2b 55 7a  2a da a3 14 c2 f8 56 18   1··6;+Uz *·····V·
0060  9f 3c 6f fa 8e d1 f5 53  16 a2 fc f1 ee aa 8b 65   ·<o····S ·······e
0070  3f 8a 20 e9 ea 8d 95 f7  8f e9 06 f9 a8 9d f7 fd   ?· ····· ········
0080  d2 7c f6 41 6f 8a ba d1  34 15 d1 fe bc 4f bc 52   ·|·Ao··· 4····O·R
0090  b9 c9 76 41 79 38 f3 1b  10 ac 77 ba ac c3 b7 9e   ··vAy8·· ··w·····
00a0  8c 8e d9 c6 a8 e6 b6 63  fc 67 ef cc 35 c2 93 ff   ·······c ·g··5···
```

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



```
No.  Time      Source           Destination      Protocol  Length Info
 436 3.429118  193.167.100.100  193.167.0.100    QUIC          69 Protected Payload (KP0), DCID=3a72c3f99f03ad
 437 3.429370  193.167.0.100    193.167.100.100  QUIC          76 Protected Payload (KP0), DCID=b95d3eff, PKN:
 438 3.429460  193.167.100.100  193.167.0.100    HTTP3       1282 Protected Payload (KP0), DCID=3a72c3f99f03ad
 439 3.429801  193.167.100.100  193.167.0.100    HTTP3       1282 Protected Payload (KP0), DCID=3a72c3f99f03ad
 440 3.430143  193.167.100.100  193.167.0.100    HTTP3       1282 Protected Payload (KP0), DCID=3a72c3f99f03ad
 441 3.430485  193.167.100.100  193.167.0.100    HTTP3       1282 Protected Payload (KP0), DCID=3a72c3f99f03ad
 442 3.430735  193.167.0.100    193.167.100.100  QUIC          61 Protected Payload (KP0), DCID=b95d3eff, PKN:
 443 3.430827  193.167.100.100  193.167.0.100    HTTP3       1282 Protected Payload (KP0), DCID=3a72c3f99f03ad
 444 3.431169  193.167.100.100  193.167.0.100    HTTP3       1282 Protected Payload (KP0), DCID=3a72c3f99f03ad
 445 3.431511  193.167.100.100  193.167.0.100    HTTP3       1282 Protected Payload (KP0), DCID=3a72c3f99f03ad
 446 3.431853  193.167.100.100  193.167.0.100    HTTP3       1282 Protected Payload (KP0), DCID=3a72c3f99f03ad
 447 3.432194  193.167.100.100  193.167.0.100    HTTP3       1282 Protected Payload (KP0), DCID=3a72c3f99f03ad
```

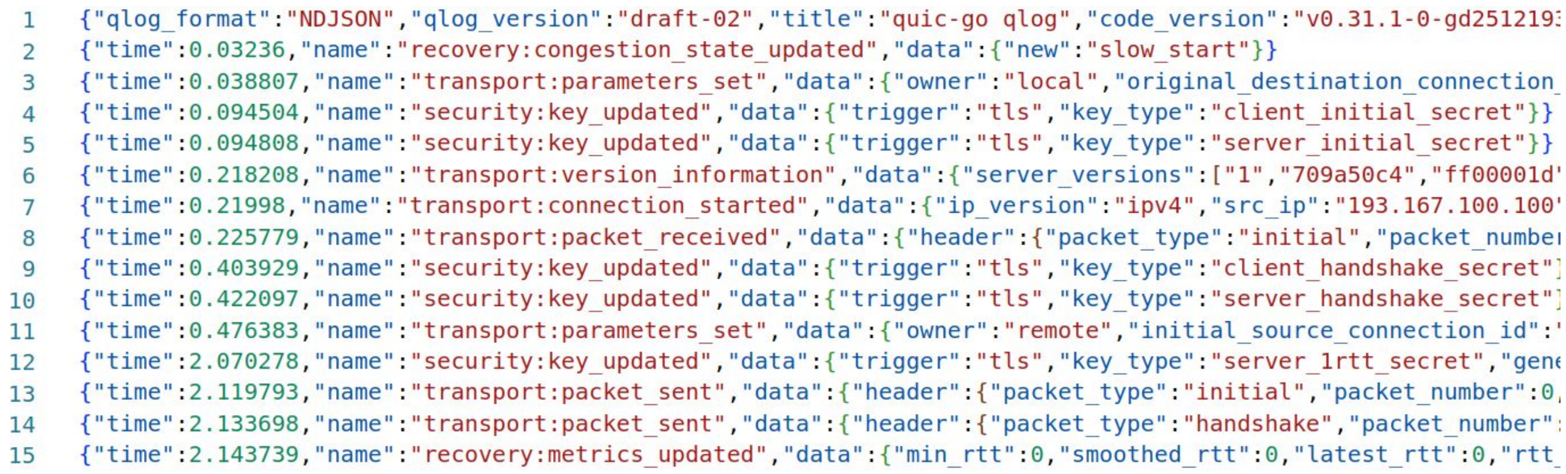
```
▸ Frame 444: 1282 bytes on wire (10256 bits), 1282 bytes captured (102
▸ Point-to-Point Protocol
▸ Internet Protocol Version 4, Src: 193.167.100.100, Dst: 193.167.0.10
▸ User Datagram Protocol, Src Port: 443, Dst Port: 55094
▾ QUIC IETF
  ▸ QUIC Connection information
    [Packet Length: 1252]
  ▸ QUIC Short Header DCID=3a72c3f99f03ad9e PKN=364
  ▾ STREAM id=0 fin=0 off=316175 len=1219 dir=Bidirectional origin=Cli
    ▸ Frame Type: STREAM (0x000000000000000c)
```

```
0c 00 80 04 d3 0f 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
```

Decrypted payload

Now let’s look at what happened under the hood

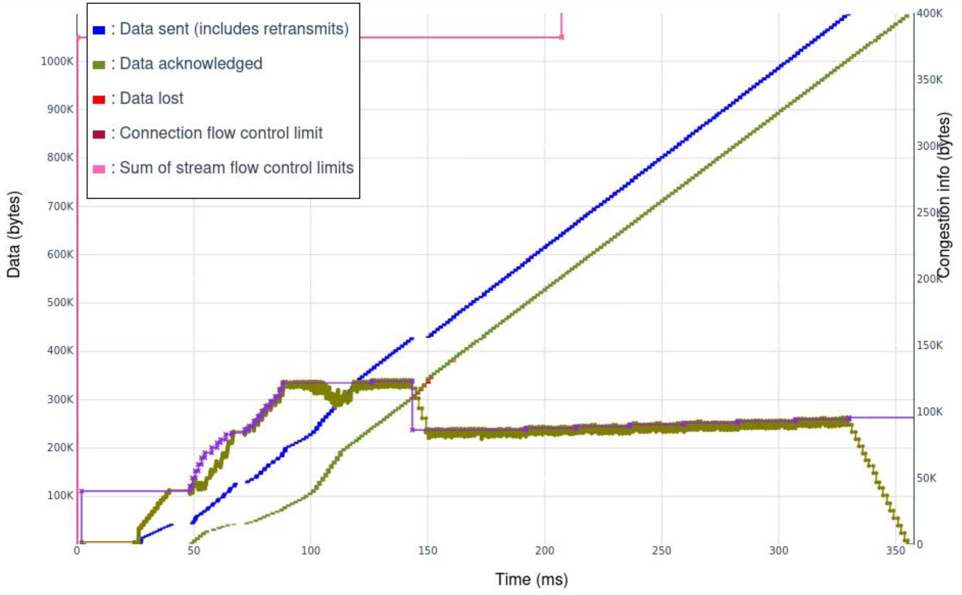
- Structured endpoint logging with [qlog]
- < qvis > visualization tools for QUIC and HTTP/3
- Great FOSDEM talk about qlog and qvis:
  https://archive.fosdem.org/2021/schedule/event/webperf_quic_http3_qlog_qvis/

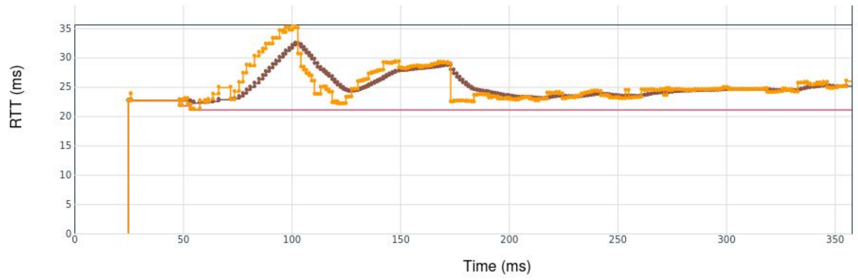


```
{"qlog_format":"NDJSON","qlog_version":"draft-02","title":"quic-go qlog","code_version":"v0.31.1-0-gd251219
{"time":0.03236,"name":"recovery:congestion_state_updated","data":{"new":"slow_start"}}
{"time":0.038807,"name":"transport:parameters_set","data":{"owner":"local","original_destination_connection_
{"time":0.094504,"name":"security:key_updated","data":{"trigger":"tls","key_type":"client_initial_secret"}}
{"time":0.094808,"name":"security:key_updated","data":{"trigger":"tls","key_type":"server_initial_secret"}}
{"time":0.218208,"name":"transport:version_information","data":{"server_versions":["1","709a50c4","ff00001d'
{"time":0.21998,"name":"transport:connection_started","data":{"ip_version":"ipv4","src_ip":"193.167.100.100'
{"time":0.225779,"name":"transport:packet_received","data":{"header":{"packet_type":"initial","packet_numbe
{"time":0.403929,"name":"security:key_updated","data":{"trigger":"tls","key_type":"client_handshake_secret"
{"time":0.422097,"name":"security:key_updated","data":{"trigger":"tls","key_type":"server_handshake_secret"
{"time":0.476383,"name":"transport:parameters_set","data":{"owner":"remote","initial_source_connection_id":'
{"time":2.070278,"name":"security:key_updated","data":{"trigger":"tls","key_type":"server_1rtt_secret","gene
{"time":2.119793,"name":"transport:packet_sent","data":{"header":{"packet_type":"initial","packet_number":0,
{"time":2.133698,"name":"transport:packet_sent","data":{"header":{"packet_type":"handshake","packet_number":
{"time":2.143739,"name":"recovery:metrics_updated","data":{"min_rtt":0,"smoothed_rtt":0,"latest_rtt":0,"rtt_
```

Now let's look at what happened under the hood

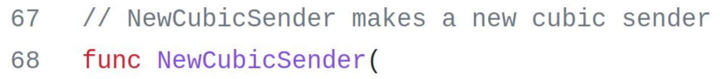
- <qvis> tool: https://qvis.quictools.info/





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



```
// NewCubicSender makes a new cubic sender
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.RTTStats,
        initialMaxDatagramSize protocol.ByteCount,
        reno bool,
        tracer logging.ConnectionTracer,
) *cubicSender {
```

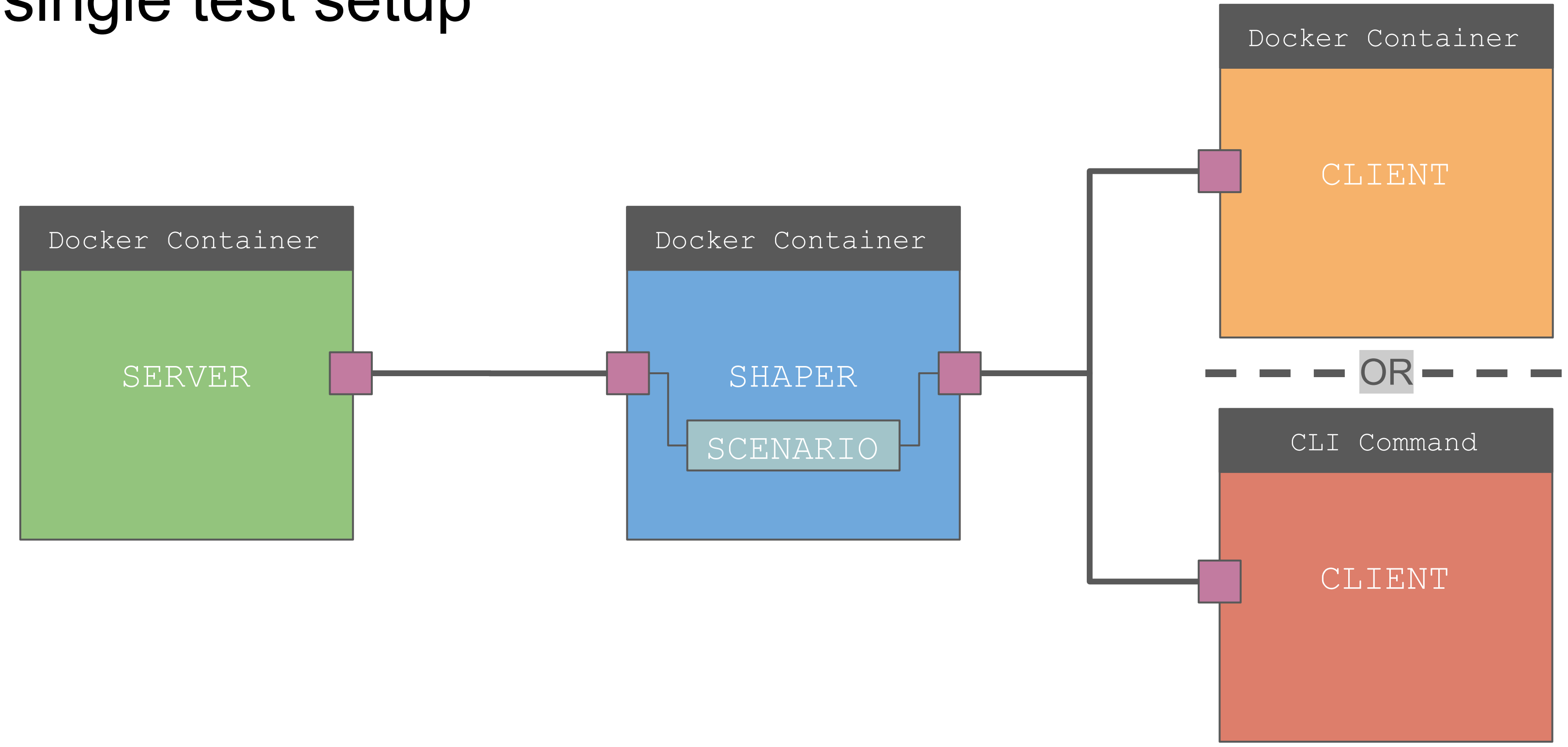
⇒ Testing them all – or even a hand-picked selection – **takes time**

VEGVISIR

https://github.com/JorisHerbots/vegvisir/

Vegvisir V2.0.0 | Python ≥3.7 | License Apache 2.0 | Linux X86-64

A single test setup



Steering Vegvisir with JSON configurations

📦 Implementation Configuration

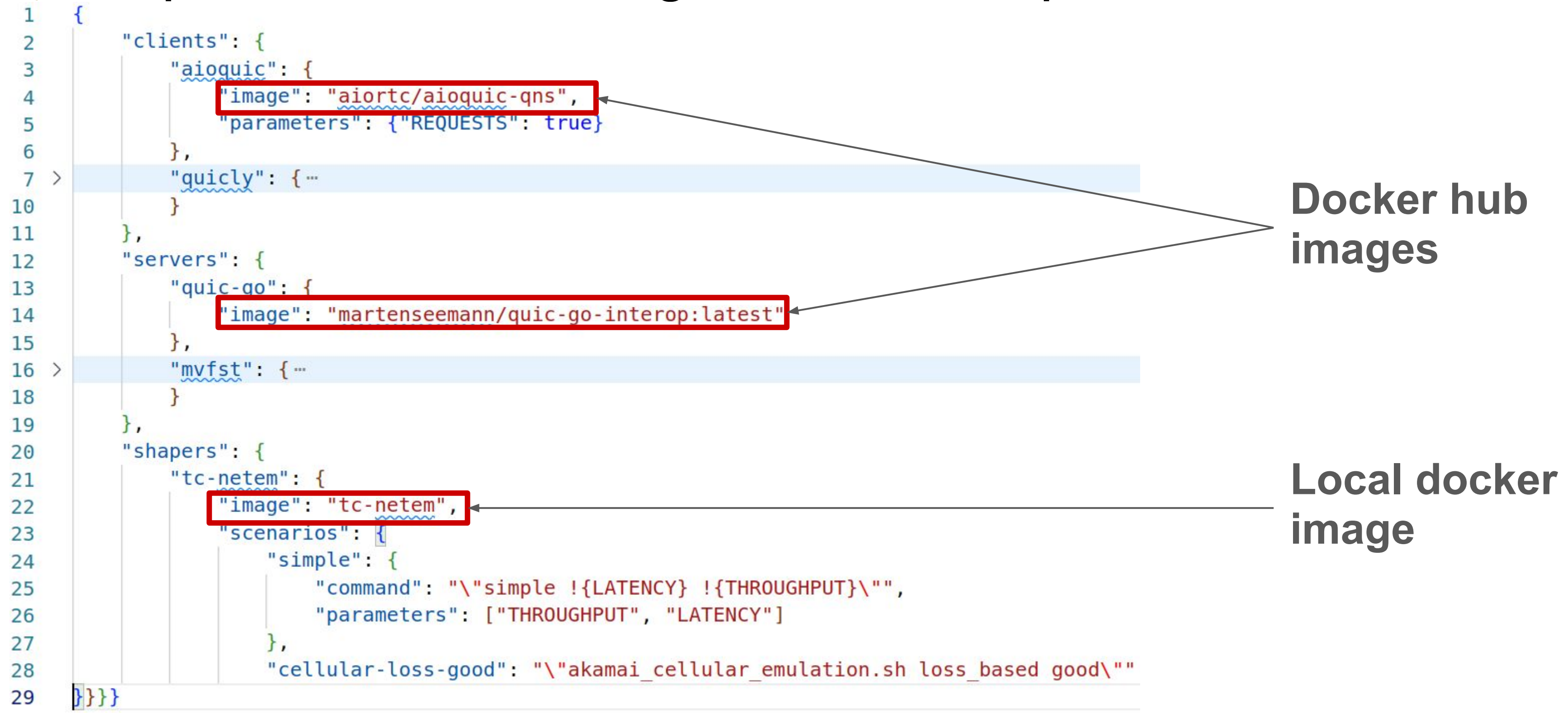
- Defines **what** entities are available
- Declares **parameters**

⚙️ Experiment Configuration

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

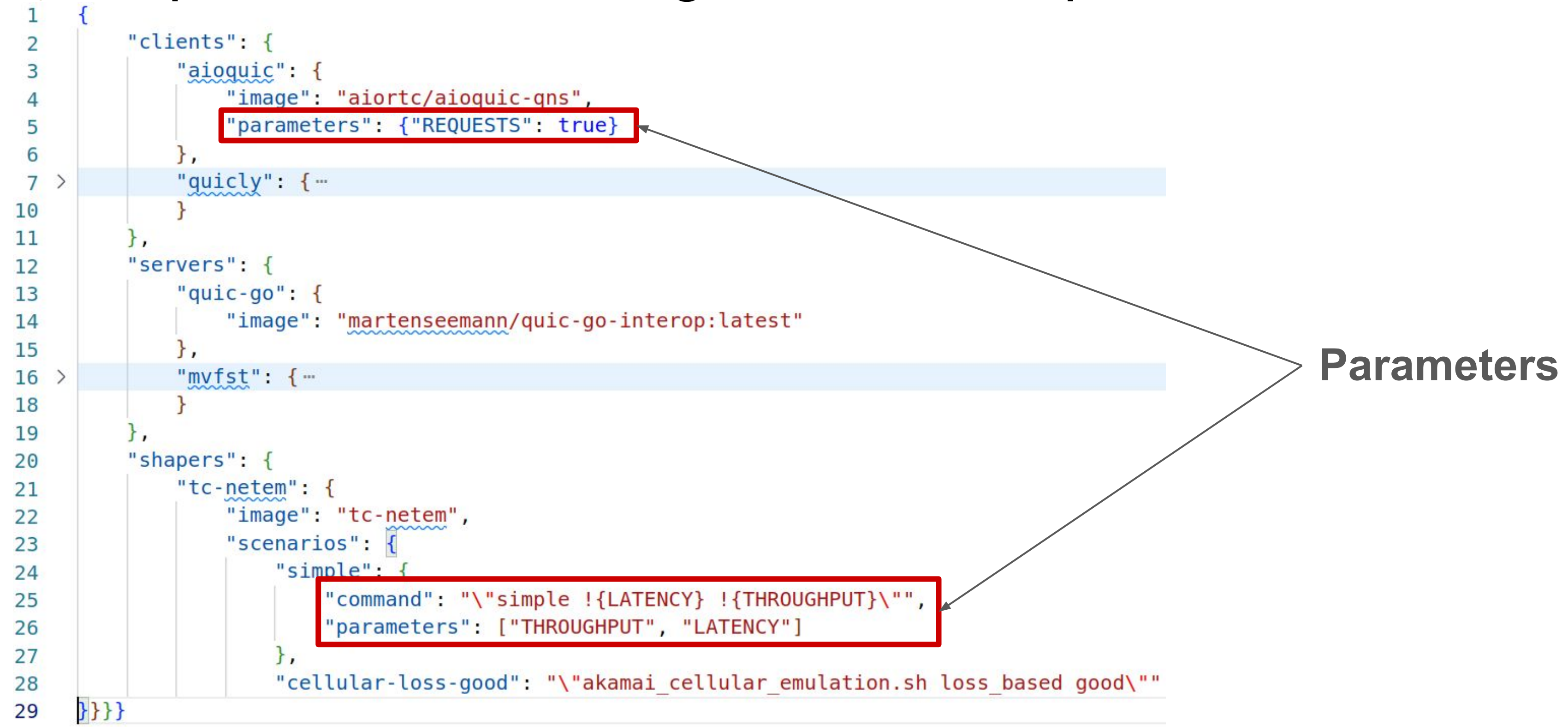
Loose coupling

📦 Implementation configuration example



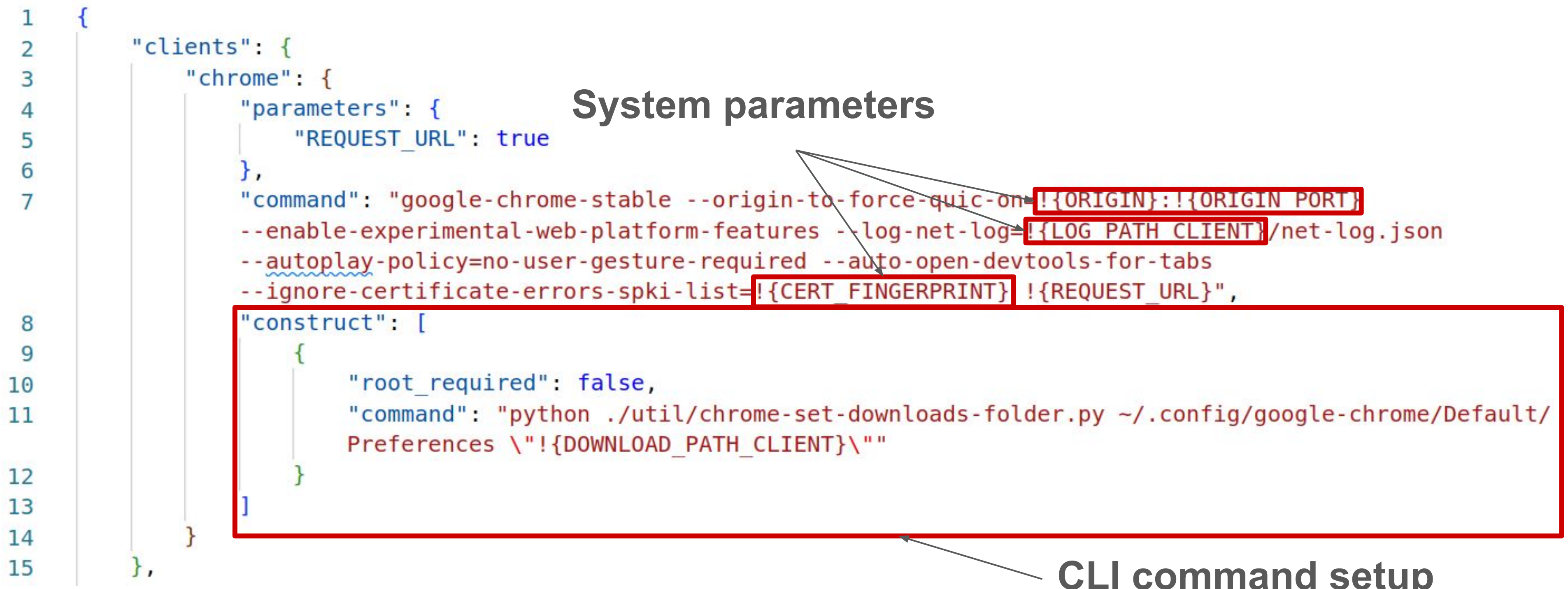
```
{
    "clients": {
        "aioquic": {
            "image": "aiortc/aioquic-qns",
            "parameters": {"REQUESTS": true}
        },
        "quicly": {⋯
        }
    },
    "servers": {
        "quic-go": {
            "image": "martenseemann/quic-go-interop:latest"
        },
        "mvfst": {⋯
        }
    },
    "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 example



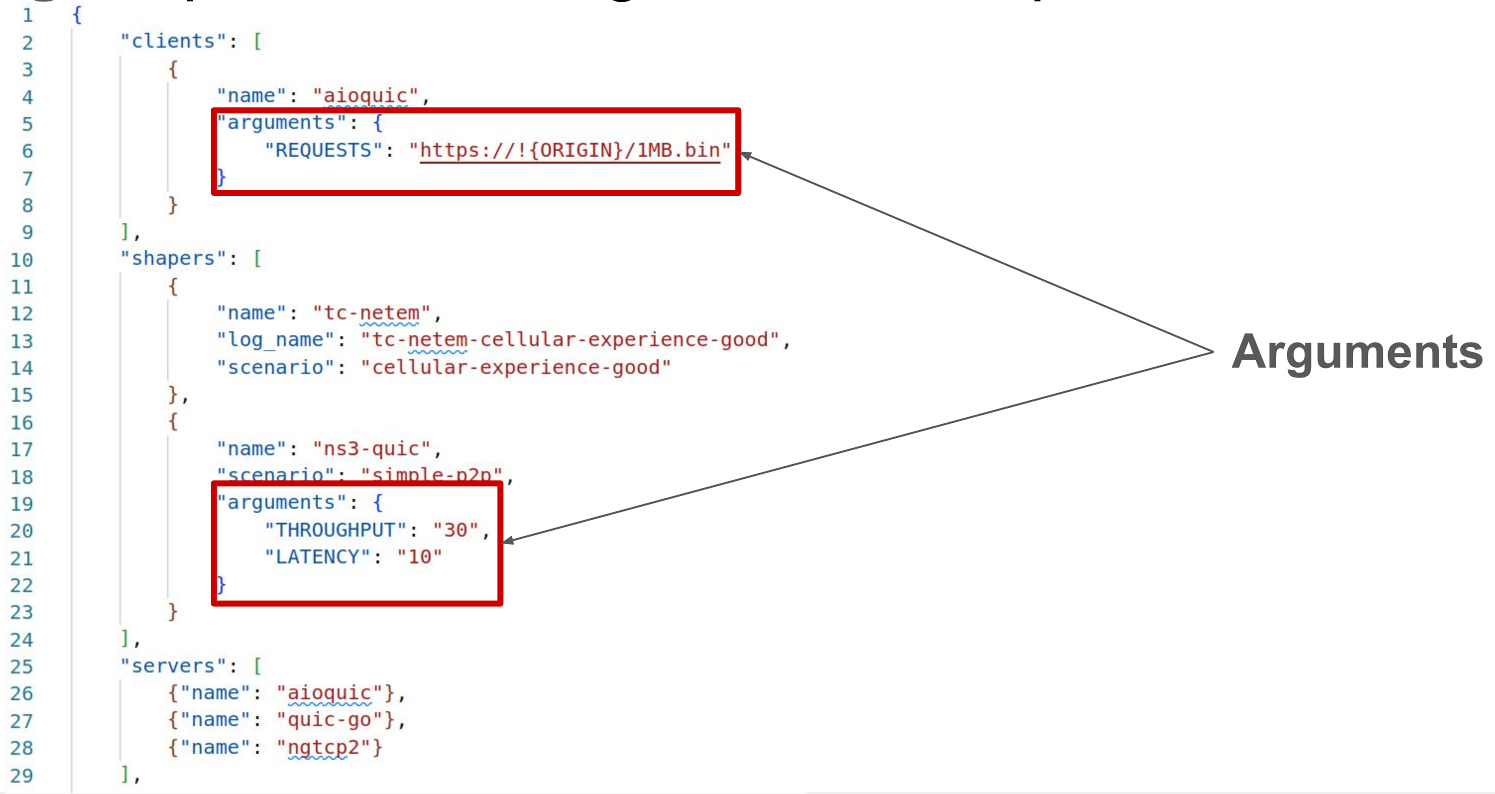
```
{
    "clients": {
        "aioquic": {
            "image": "aiortc/aioquic-qns",
            "parameters": {"REQUESTS": true}
        },
        "quicly": {…
        }
    },
    "servers": {
        "quic-go": {
            "image": "martenseemann/quic-go-interop:latest"
        },
        "mvfst": {…
        }
    },
    "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



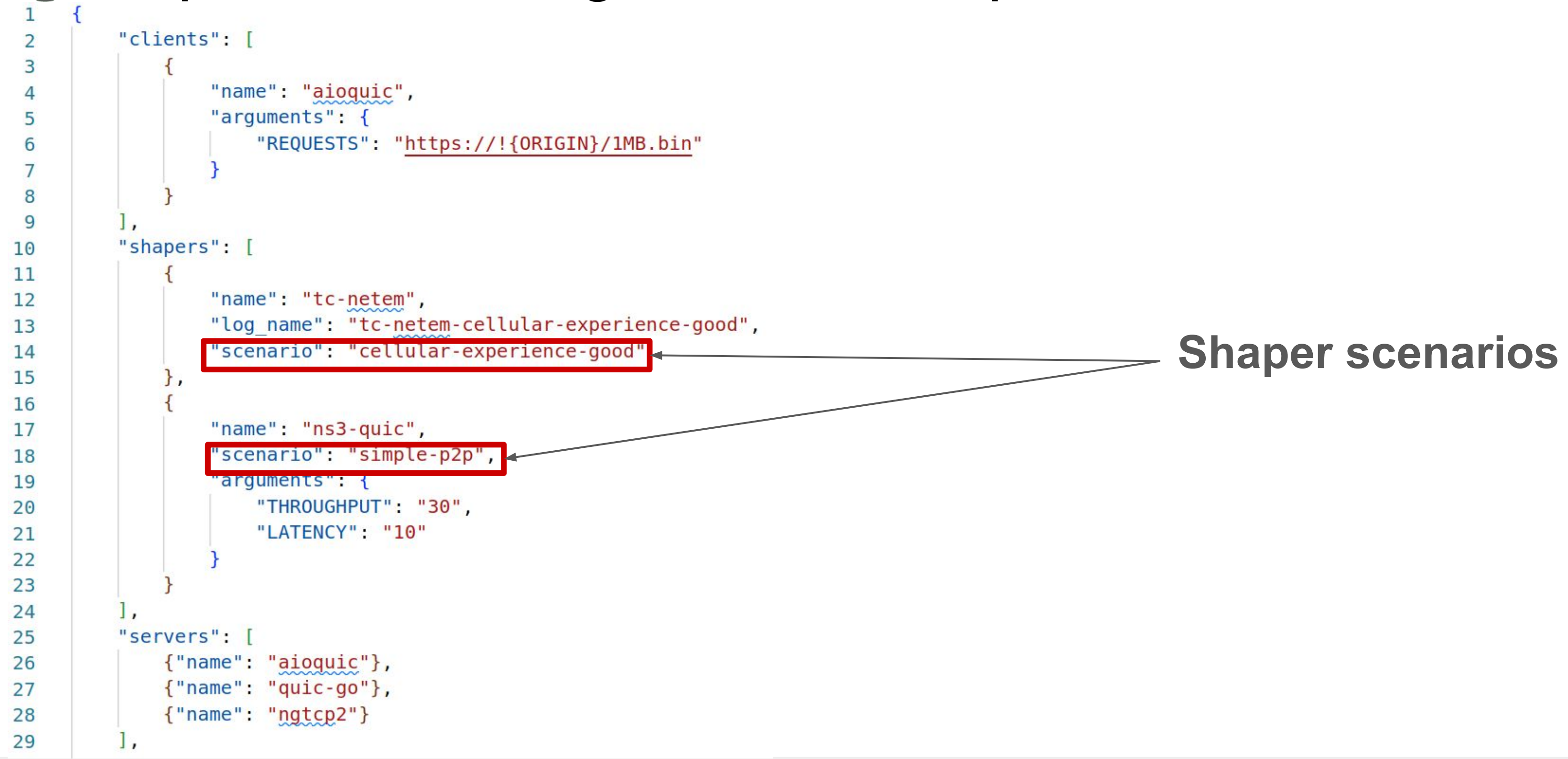
```
{
    "clients": {
        "chrome": {
            "parameters": {
                "REQUEST_URL": true
            },
            "command": "google-chrome-stable --origin-to-force-quic-on=!{ORIGIN}:!{ORIGIN_PORT} --enable-experimental-web-platform-features --log-net-log=!{LOG_PATH_CLIENT}/net-log.json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

Experiment configuration example



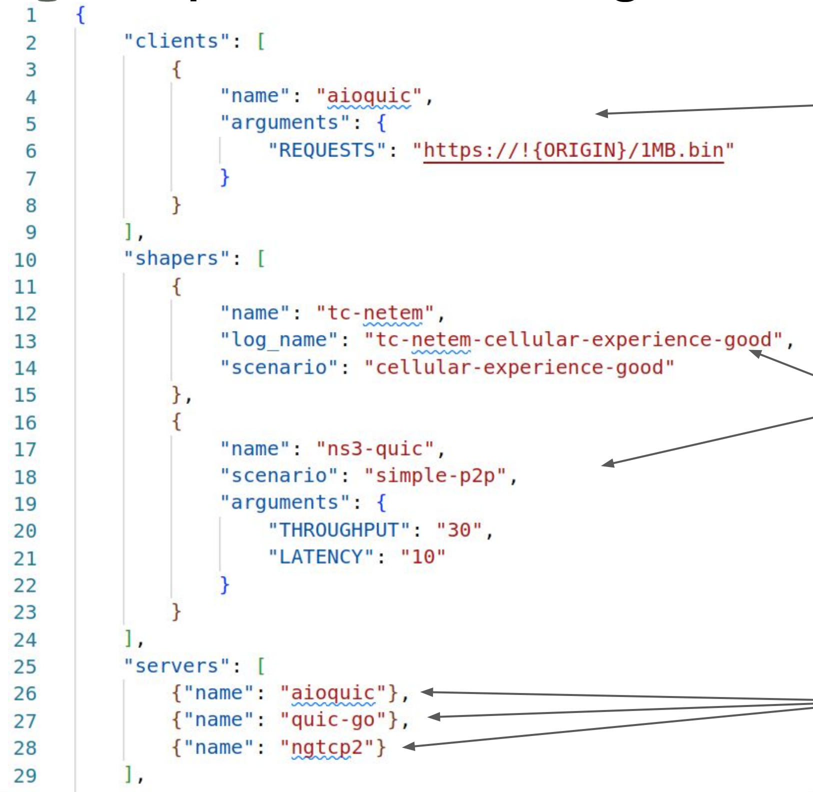
```
{
    "clients": [
        {
            "name": "aioquic",
            "arguments": {
                "REQUESTS": "https://!{ORIGIN}/1MB.bin"
            }
        }
    ],
    "shapers": [
        {
            "name": "tc-netem",
            "log_name": "tc-netem-cellular-experience-good",
  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

Experiment configuration example



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

Experiment configuration example



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

1 client

✖

2 shapers

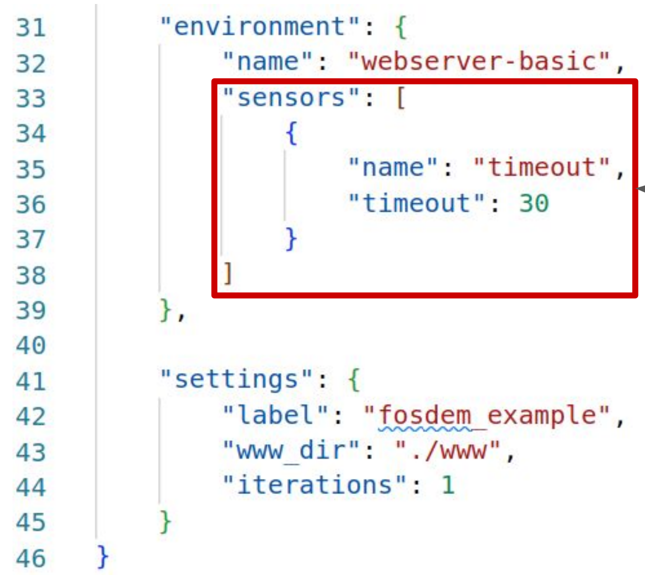
✖

3 servers

=

6 tests

Experiment configuration example



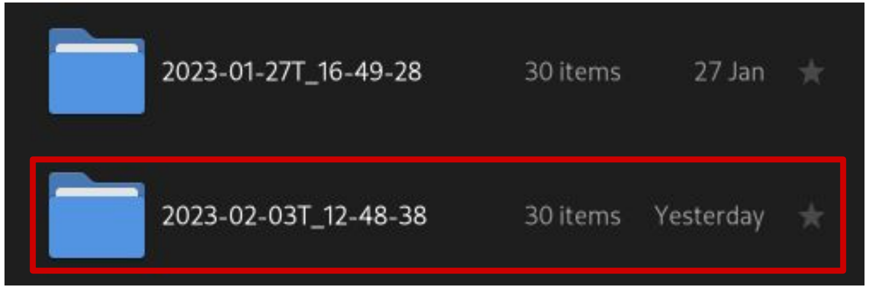
```
    "environment": {
        "name": "webserver-basic",
        "sensors": [
            {
                "name": "timeout",
                "timeout": 30
            }
        ]
    },

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

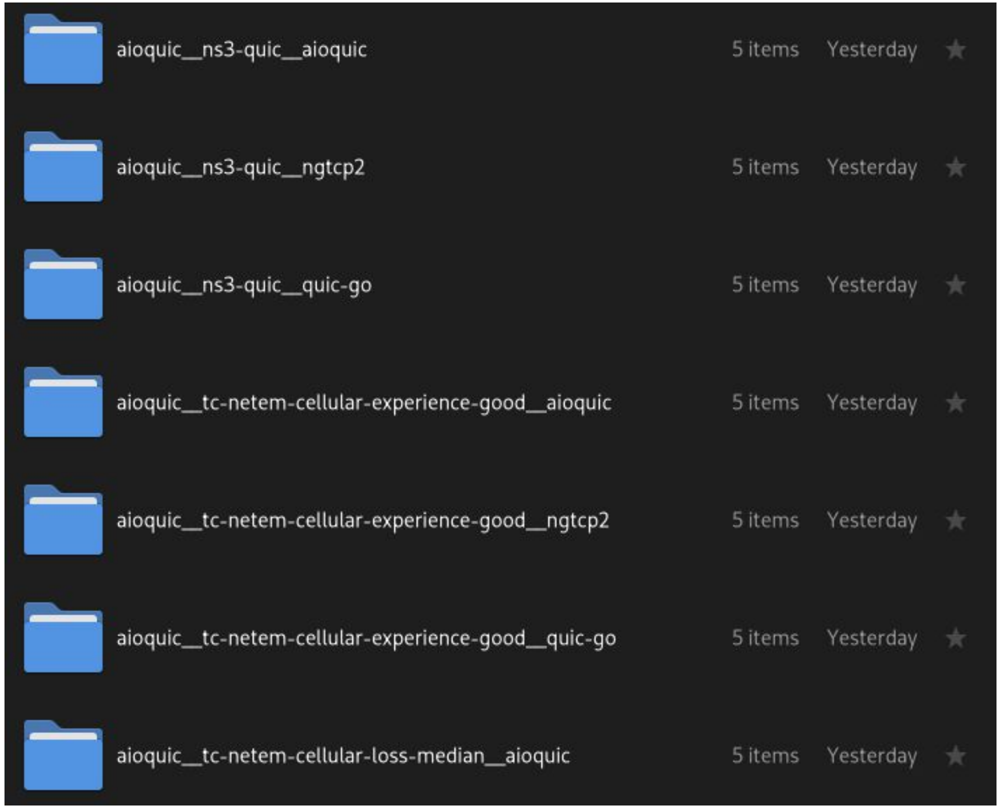
Sensor config

Vegvisir

Experiment output


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

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


aioquic__ns3-quic__aioquic 5 items Yesterday

aioquic__ns3-quic__ngtcp2 5 items Yesterday

aioquic__ns3-quic__quic-go 5 items Yesterday

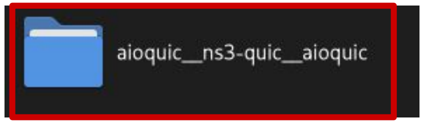
aioquic__tc-netem-cellular-experience-good__aioquic 5 items Yesterday

aioquic__tc-netem-cellular-experience-good__ngtcp2 5 items Yesterday

aioquic__tc-netem-cellular-experience-good__quic-go 5 items Yesterday

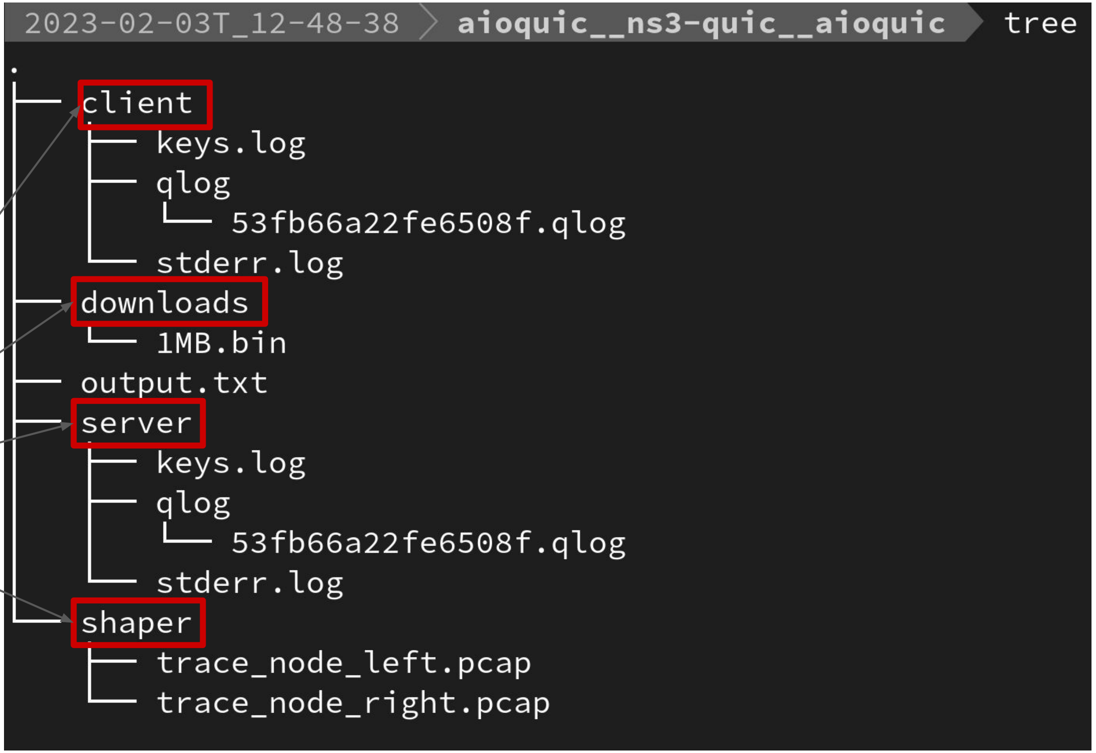
aioquic__tc-netem-cellular-loss-median__aioquic 5 items Yesterday

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



```
2023-02-03T_12-48-38 〉 aioquic__ns3-quic__aioquic 〉 tree
.
├── client
│   ├── keys.log
│   ├── qlog
│   │   └── 53fb66a22fe6508f.qlog
│   └── stderr.log
├── downloads
│   └── 1MB.bin
├── output.txt
├── server
│   ├── keys.log
│   ├── qlog
│   │   └── 53fb66a22fe6508f.qlog
│   └── stderr.log
└── shaper
    ├── trace_node_left.pcap
    └── trace_node_right.pcap
```

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!