VEGVISIR

TESTING FRAMEWORK FOR HTTP/3 MEDIA STREAMING

Joris Herbots, Mike Vandersanden, Peter Quax, Wim Lamotte

UHASSELT
KNOWLEDGE IN ACTION

Assessing media streaming performance traditionally requires the presence of **reproducible network conditions** and a **heterogeneous dataset of media materials**. Setting up such experiments represents a complex challenge in itself. This challenge becomes even more complex when we consider the new **QUIC transport protocol**, which has many **tunable features**, yet is difficult to analyze due to its inherent encrypted nature. Vegvisir aims to solve these aforementioned challenges by providing an **open-source automated testing framework** for orchestrating **media streaming experiments over HTTP/3**.

☆ us on GitHub!

Implementation Configuration

- Defines **what** entities are available
- Entities belong to the **client**, **shaper** or **server** group
- **Parameters** allow for dynamic and rapid testing

```
{
  "clients": {
    "chrome": {
      "command": "chrome --origin-to-force-quic-on=!{ORIGIN} !{REQUESTS}",
      "parameters": {
        "REQUEST_URL": true
}}},
  "servers": {
    "aioquic": {
      "image": "aiortc/aioquic-qns"
    },
    "quic-go": {
      "image": "martenseemann/quic-go-interop:latest"
}}}
```

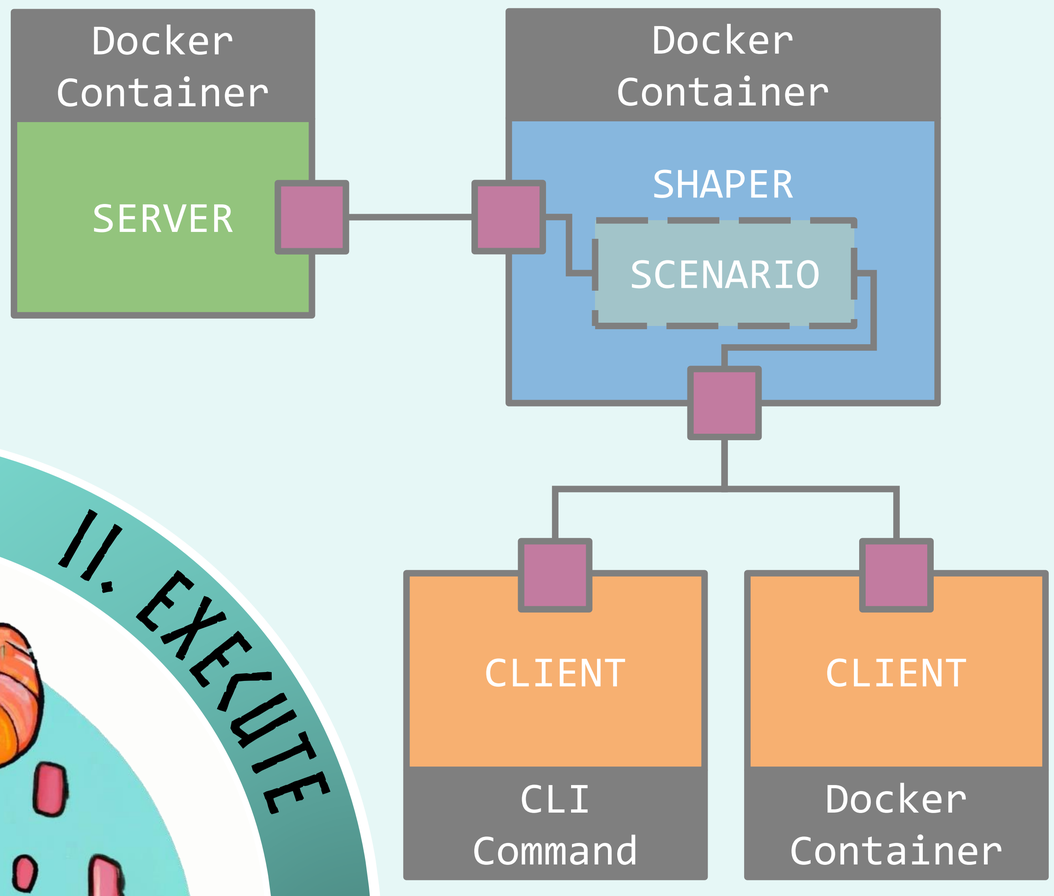
Experiment Configuration

- Defines picked entities and **how** they behave
- Provides **arguments**
- Configures **sensors** (see III)

```
{
  "clients": [
    {
      "name": "chrome",
      "arguments": {
        "REQUESTS": "https://!{ORIGIN}/1MB.bin"
      }
    }
  ]
}
```

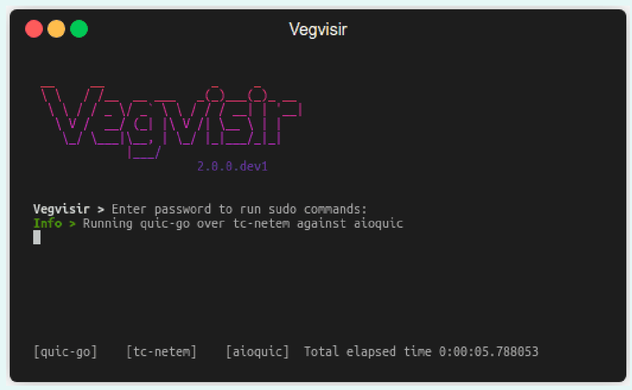
Experiment Execution Engine

- An experiment contains **one or more test(s)**
  - `|tests| = |clients| x |shapers| x |servers|`
- Tests execute **sequentially**



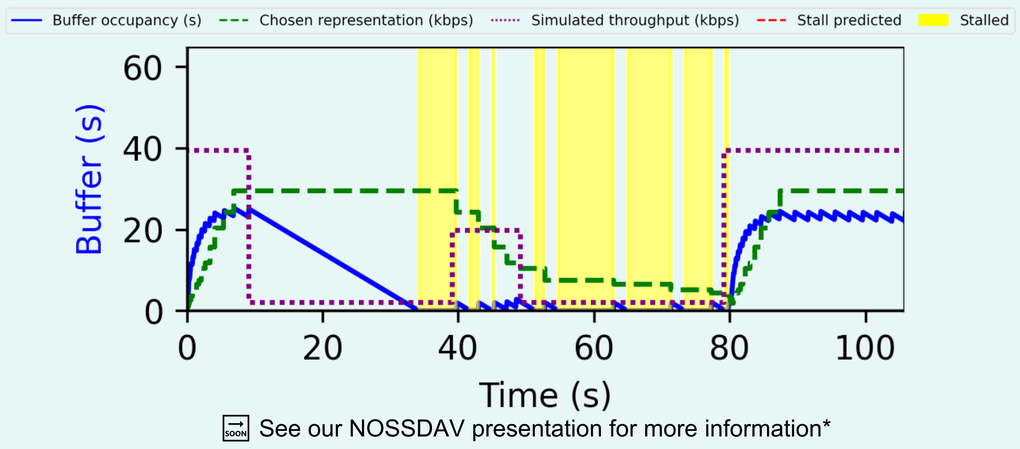
Capture experiment output

```
2023-02-03T_12-48-38 〉aioquic__ns3-quic__aioquic
.
├── 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
```



Example of automated analysis

- `post_run_hook` using matplotlib
- Client metrics converted to a graph



See our NOSSDAV presentation for more information*

Sensors monitor test conditions

- Currently available:
  - Generic timeout sensor
  - Browser file download sensor
- Ability to create **custom sensors** for your experiment

Hooks allow for customization

- Broad applicability means having little knowledge about experiments
- Program **custom behavior**:
  - Pre-test `pre_run_hook` to **prime environments**
  - Post-test `post_run_hook` to **automate analysis**

* https://doi.org/10.1145/3592473.3592563
Joris Herbots (BOF19OWB07) and Mike Vandersanden (BOF22OWB17) are Ph.D. candidates at Hasselt University, supported by the Special Research Fund (BOF).