QUIC Tracker

An active test tool for QUIC

Maxime Piraux

May 7, 2019

This work was partially supported by funding from the Walloon Government (DGO6) within the MQUIC project

Complexity of the QUIC specification

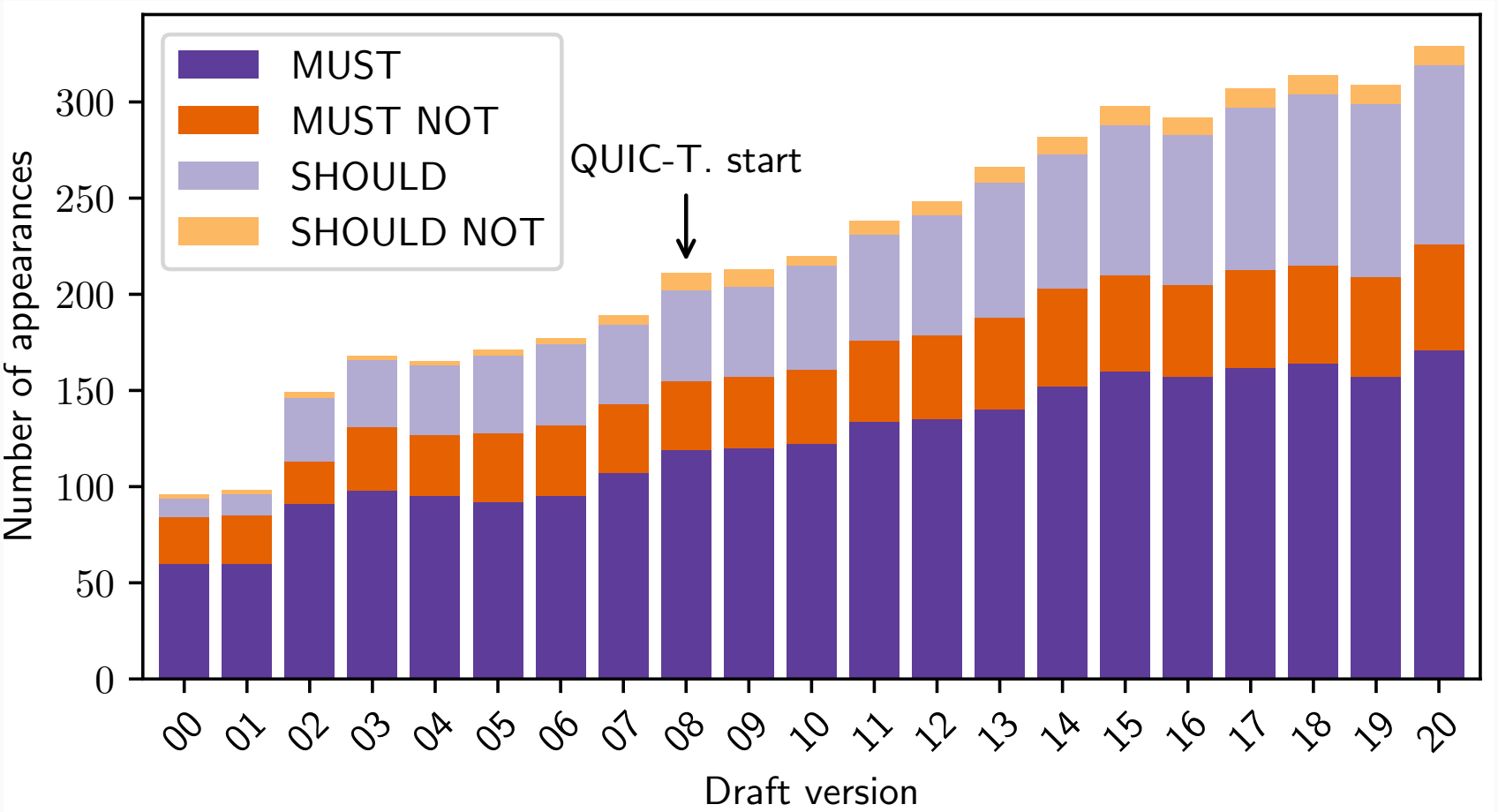


Figure 1: Evolution of keywords in `draft-ietf-quic-transport`.

QUIC implementations are evolving rapidly

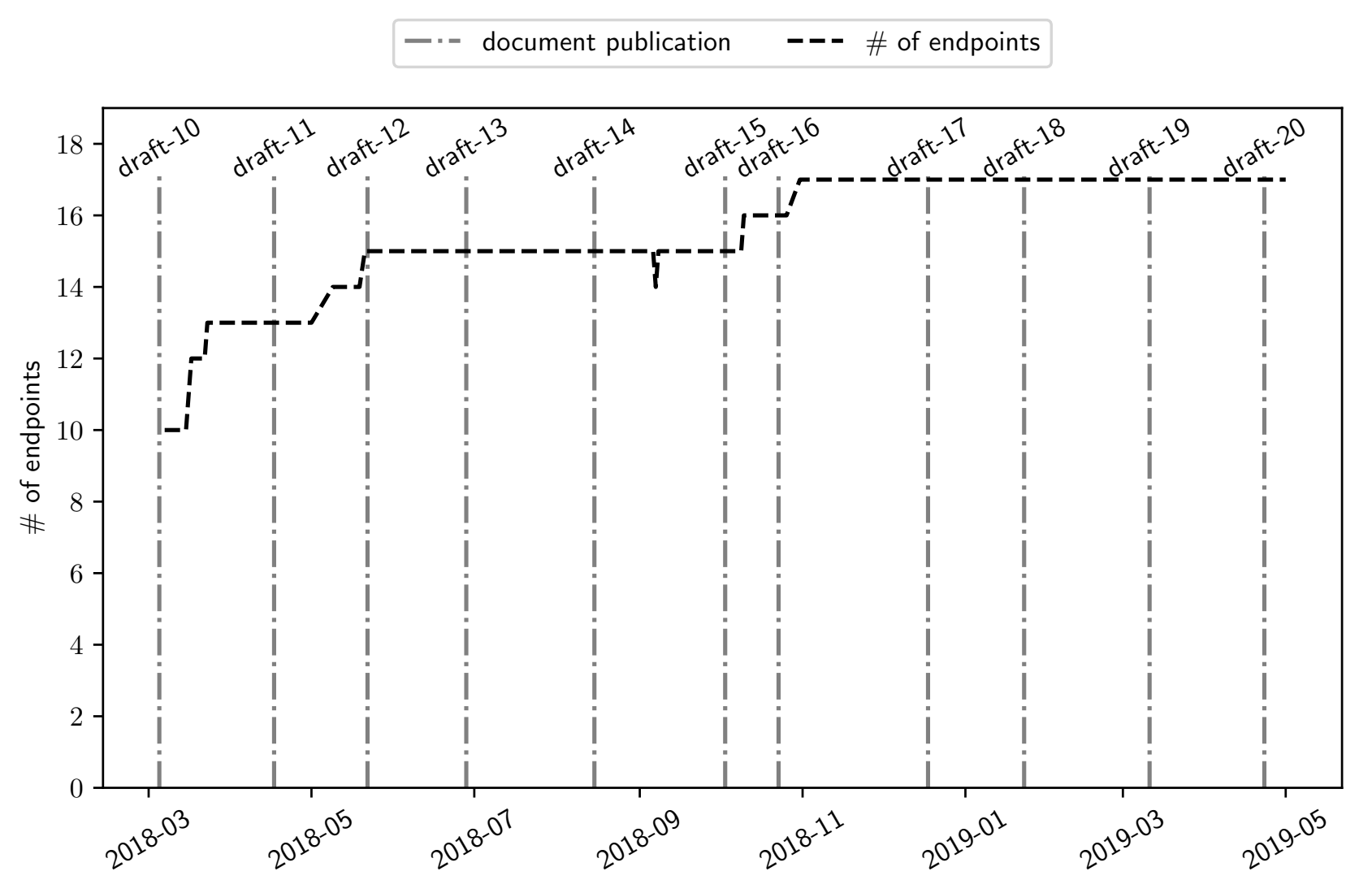


Figure 2: Evolution of QUIC versions.

QUIC implementations are evolving rapidly

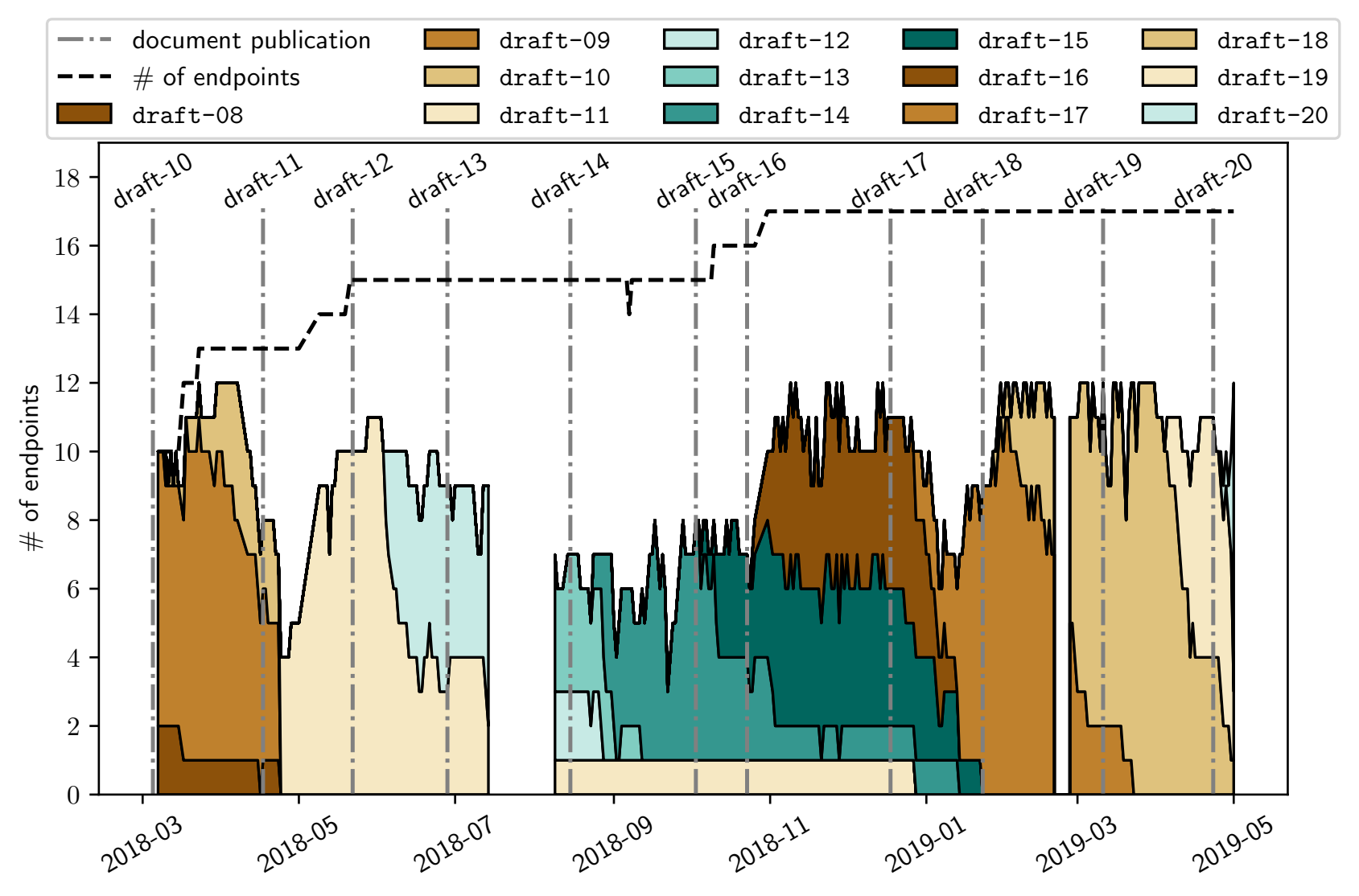


Figure 2: Evolution of QUIC versions.

Contributing to the QUIC effort

- We propose an active test tool called **QUIC Tracker**.
- It exchanges packets with server implementations to test them.
- The tool runs daily and its results are public.

Results grid

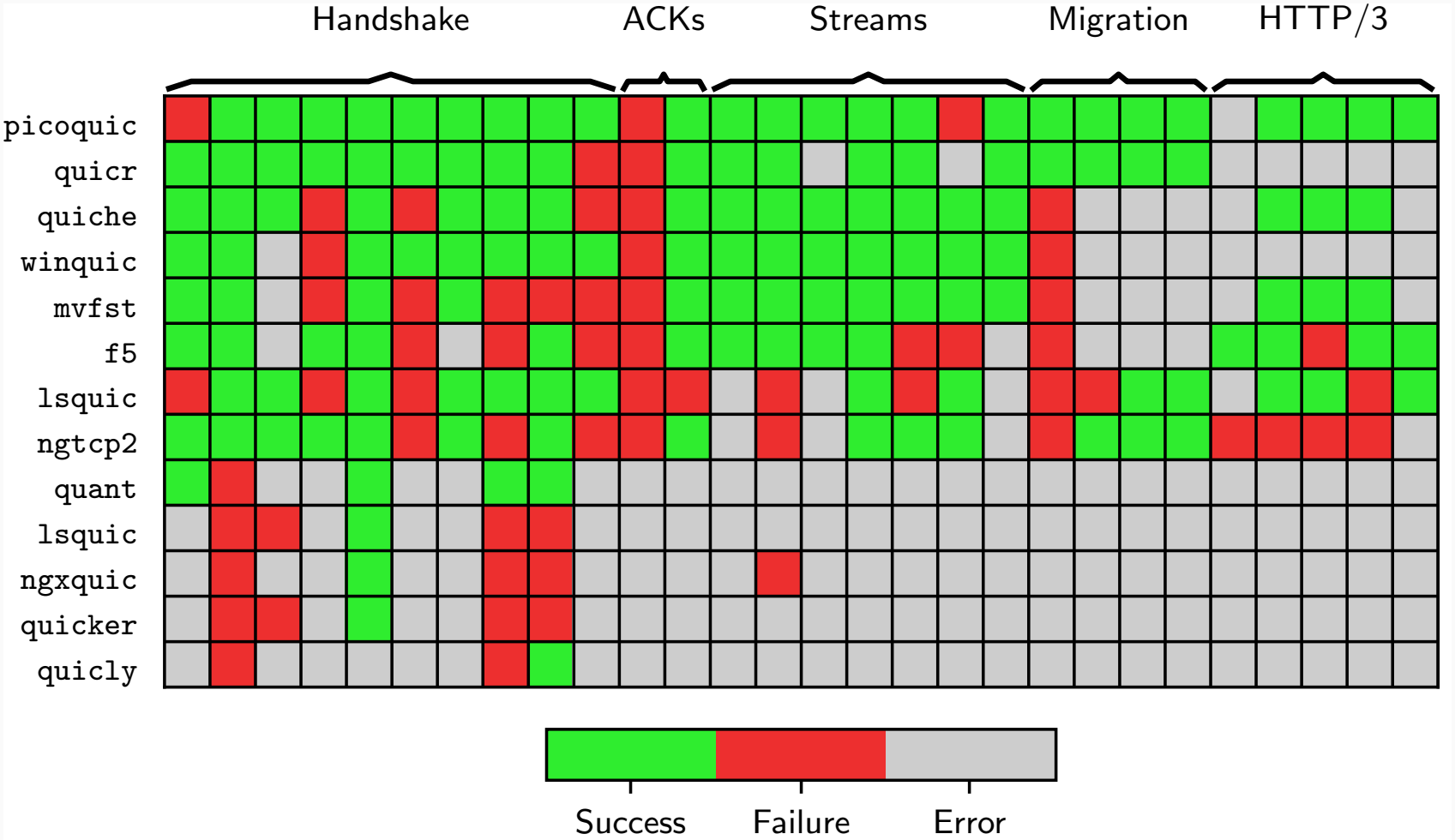


Figure 3: Results grid on the 2nd of May.

QUIC Tracker architecture

Architecture

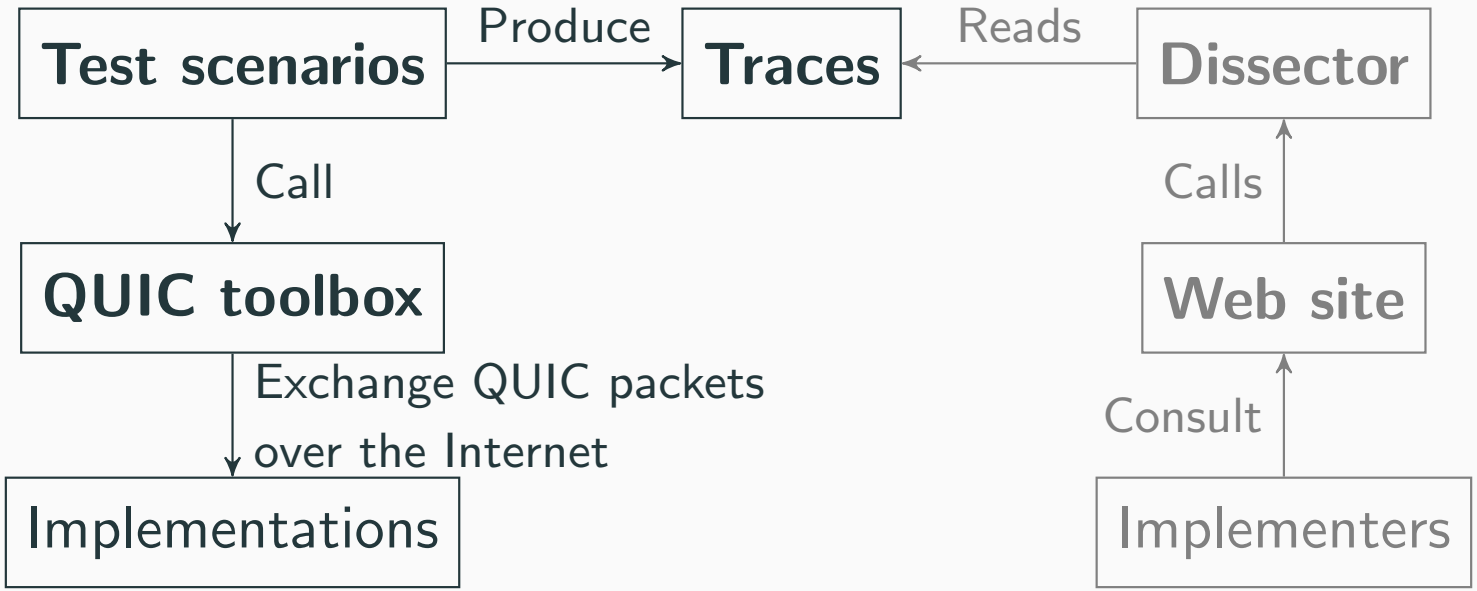
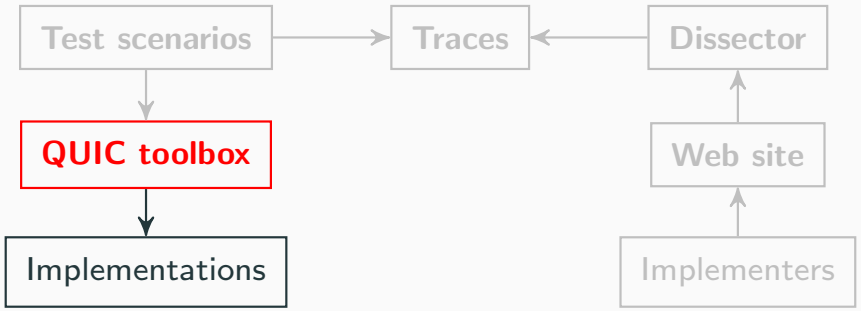


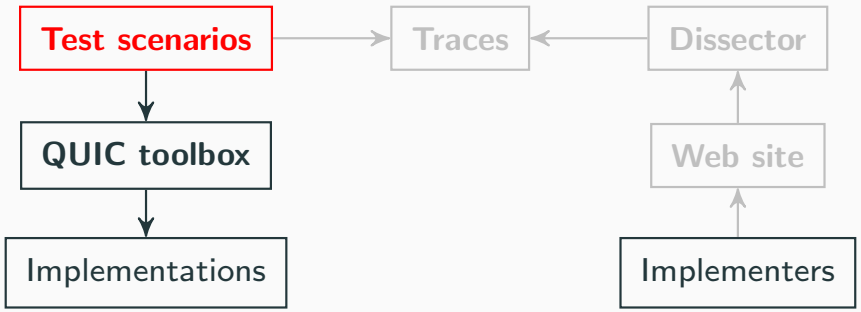
Figure 4: Tools forming QUIC Tracker.

Architecture – QUIC toolbox



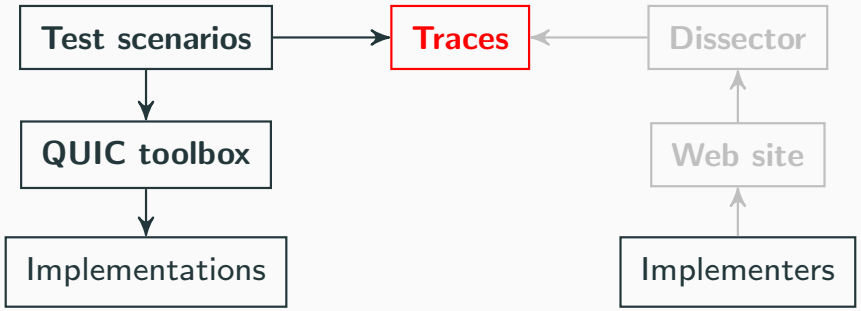
- A library built in Go to implement QUIC clients.
- It provides a high-level API to manipulate QUIC packets.
- It implements all types of connection establishment, streams, recovery, acknowledgements and HTTP/3.
- The library consists of 5000 lines of code.

Architecture – Test scenarios



- There currently exist 28 test scenarios.
- We derive rules that should not be violated from the specification.
- Each test is executed in a separate connection.
- Each test targets a particular feature of QUIC.
- A test is 56-line long in average.

Architecture – Traces



- We defined a JSON trace format common to all tests.
- A trace contains an error code summarising its outcome.
- Scenario-specific data can be embedded, e.g. list of supported versions.
- The exchanged packets are recorded inside the trace.

Recent improvements

Recap since EPIQ'18

- `draft-20` is supported.
- We are phasing `hq`-based tests out in favour of HTTP/3.
- Test scenarios development has been further simplified.
- 5 more tests were added.
  - Spin bit
  - IPv4 $\rightarrow$ IPv6 migration
  - Two HTTP/3 greasing tests
  - Client flow control violation

Bringing declarative programming to QUIC Tracker

- Declarative programming seems a natural fit for test suites.
- E.g. `quic-go` uses `Gomega` for its internal test suite.

Bringing declarative programming to QUIC Tracker

```
var firstCrypto bool
for _, f := range p.GetAll(qt.CryptoType) {
    if f.(*qt.CryptoFrame).Offset == 0 {
        firstCrypto = true
        break;
    }
}
```

Figure 5: Imperative programming

Bringing declarative programming to QUIC Tracker

```
var firstCrypto bool
for _, f := range p.GetAll(qt.CryptoType) {
    if f.(*qt.CryptoFrame).Offset == 0 {
        firstCrypto = true
        break;
    }
}
```

Figure 5: Imperative programming

```
firstCrypto := Expect(p.GetAll(qt.CryptoType))
    .To(ContainElement(MatchAllFields(IgnoreExtras, Fields{
     "Offset": Equal(uint64(0)),
})))
```

Figure 6: Declarative programming

Bringing declarative programming to QUIC Tracker

- Declarative programming seems a natural fit for test suites.
- E.g. `quic-go` uses `Gomega` for its internal test suite.
- QUIC Tracker is already quite simple.
- There is no clear benefit in switching to declarative programming.
- `Gomega` cannot be reused as is in QUIC Tracker.

Future prospects

Introduce a DSL for test scripting

- QUIC Tracker is not `packetdrill` (yet).
- QUIC semantics are far more complex than TCP's.
- *Can we invent a syntax rich enough to express them ?*

Test QUIC clients

- Advertisements on a popular web site can force clients to connect to the tool.
- We should be careful not to mess with the clients too much.
- How to distinguish between the clients ?

Study the usability of QUIC from different ASNs[1]

- By using QUIC Tracker from different ASNs, one could map the usability of QUIC in the Internet.
- UDP blockage or more advanced middleboxes interferences could be detected using the test scenarios.
- A mobile deployment would help the data gathering.

[1] https://github.com/QUIC-Tracker/quic-tracker/issues/11

Conduct active measurements in the wild

- Several works studied TCP deployment and configuration in the past
- Similar measurements could be conducted for QUIC
- E.g. measuring the *initial congestion window*.

Using and improving QUIC Tracker

- QUIC Tracker is a free and open-source tool.
- You are encouraged to submit ideas, suggestions and PRs.
- Its development is happening at `github.com/QUIC-Tracker`.
- `quic-tracker.info.ucl.ac.be/blog` contains a tutorial on adding new scenarios.