QUIC

> **This document is not updated**
>
> Please refer to the documentation on GitHub. https://github.com/apache/trafficserver/wiki/HTTP-3-Documentation

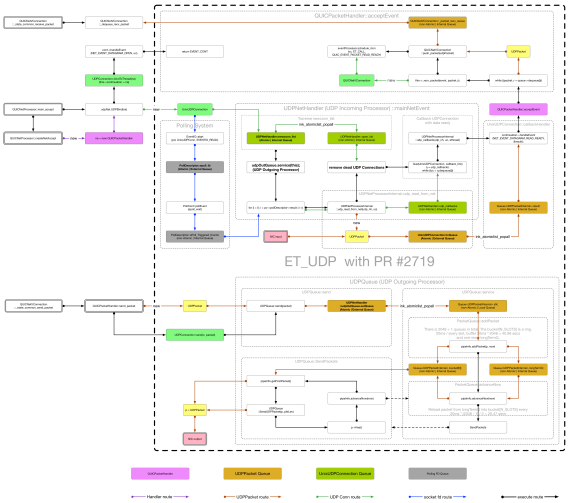
Table of contents

Goal

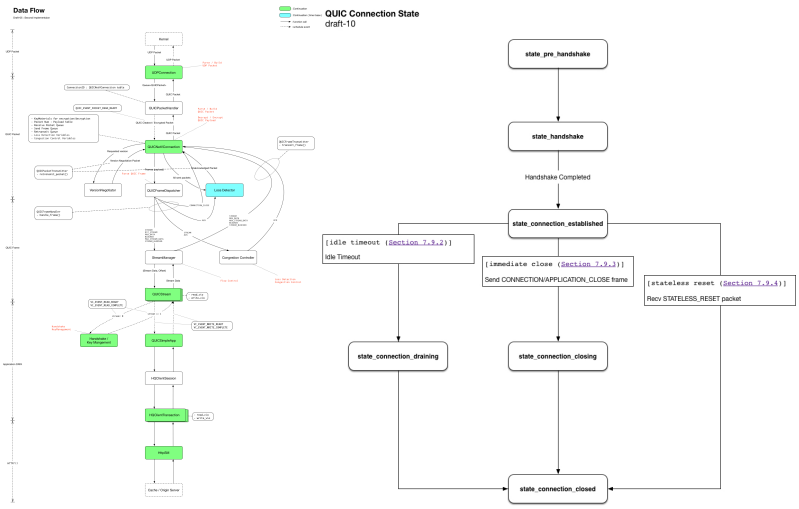
Implement IETF QUIC into ATS Core.

Design Doc

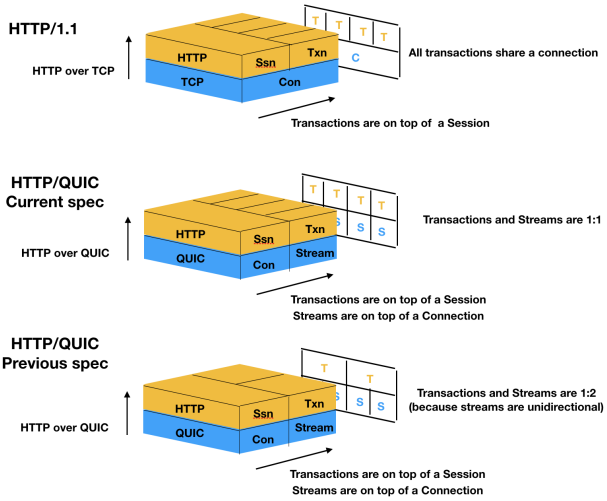
The UDP core and QUIC (It explain how the UDP core works and how the UDPPacket enters the QUIC stack.)



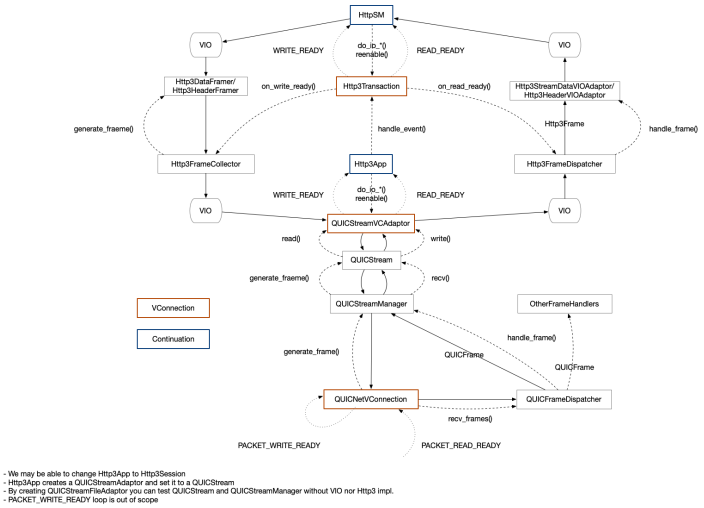
Data Flow (draft-05 : Second Implementation)



QUIC Connection/Stream - ATS Client Session/Transaction mapping (Obsolete: Not too inaccurate, but not worth referencing)

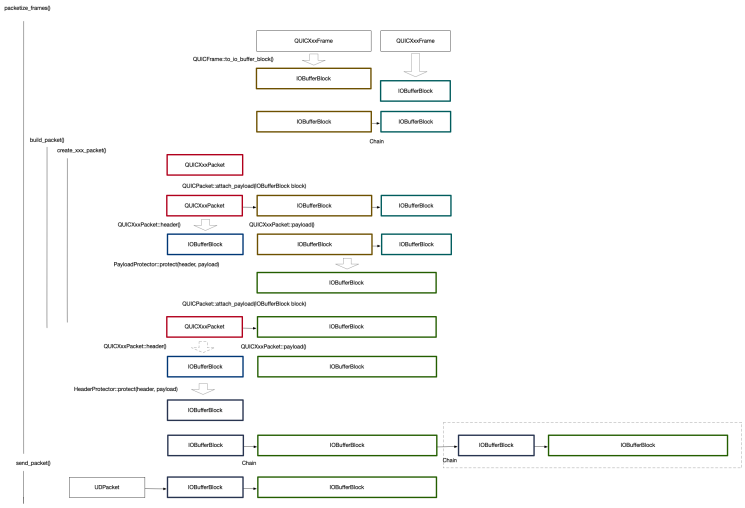


I/O between HttpSM and QUICNetVC (Jan/24/2023: Updated, but QUICFrames are handled by Quiche now)



- We may be able to change Http3App to Http3Session
- Http3App creates a QUICStreamAdaptor and set it to a QUICStream
- By creating QUICStreamFileAdaptor you can test QUICStream and QUICStreamManager without VIO nor Http3 impl.
- PACKET_WRITE_READY loop is out of scope

Packetization (Jan/24/2023: Updated, we just don't do QUIC packetization by ourselves)

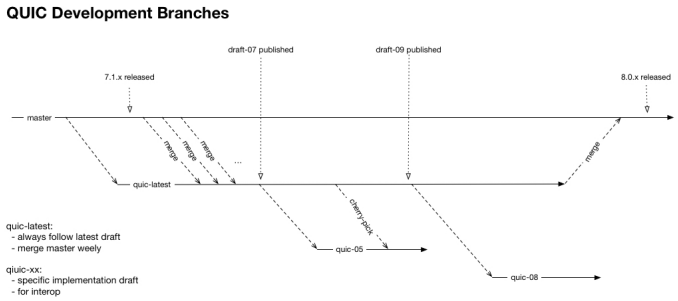


ToDo

https://github.com/apache/trafficserver/projects/8

Please label issues and pull-requests with "QUIC".

Branching Rules



Branches

Please use 10-Dev or master. Feature branch was merged and removed.

- ~~quic-latest~~ : ~~latest branch~~
- ~~master: draft-29 (currently)~~

Development Rules

~~Pull-Requests~~

~~Please send Pull-Requests to "quic-latest" branch until it merged into master branch~~

~~TDD~~

~~Use Catch as Unit-Test Framework. The header file is under tests/include.~~

How to build

(Last update: Jan/24/2023)

You have two ways to enable QUIC on ATS:

- Use Quiche library
  - This uses Quiche's QUIC implementation
- Use an SSL library that supports QUIC (i.e. BoringSSL, or OpenSSL from quictls)
  - This uses ATS's QUIC implementation

We keep ATS's native QUIC implementation for future improvement in case we need more flexibility, but our focus is currently on using Quiche.

Build Quiche (if you want to use Quiche's QUIC implementation)

Currently ATS is compatible with Quiche 0.16.0.

Please refer to the official documents for the build step. You need to enable ffi feature at minimum. qlog is also available.

https://github.com/cloudflare/quiche

Build an SSL library (if you want to use ATS's QUIC implementation)

ATS now supports 4 variation of SSL libraries. Pick one from below and build it.

BoringSSL

Official BoringSSL works without patches.

https://boringssl.googlesource.com/boringssl

These commits below work, and recent commits would probably work as well.

cbae965ca03825d517efe98cf7b8812584cab4a0 (BoringSSL API version 9)

88024df12147e56b6abd66b743ff441a0aaa09a8 (BoringSSL API version 10)

Please note that the support for BoringSSL API version 9 may be removed without notice in the future.

OpenSSL (quictls/openssl) [RECOMMENDED]

https://github.com/quictls/openssl/tree/OpenSSL_1_1_1j+quic

They also have branches based on OpenSSL 3.0 but we haven't fully supported it.

```
$ git clone --depth 1 --branch OpenSSL_1_1_1j+quic https://github.com/quictls/openssl
$ cd openssl
$ ./config --prefix=/PATH/TO/THE/OPENSSL
$ make
$ make install
```

OpenSSL (tatsuhiro-t/OpenSSL_1_1_1g-quic-draft-32) [OBSOLETE]

This is ngtcp2 developer's customized version.

https://github.com/tatsuhiro-t/openssl/tree/OpenSSL_1_1_1g-quic-draft-32

OpenSSL (akamai/master-quic-support) [INCOMPATIBLE]

This used to work, but it's incompatible now because it's based on OpenSSL master branch.

This is the branch used for https://github.com/openssl/openssl/pull/8797 .

https://github.com/akamai/openssl/tree/master-quic-support

Build ATS (10-Dev branch)

Quiche support is only available on quiche branch at the moment.

```
$ git clone --depth 1 --branch 10-Dev https://github.com/apache/trafficserver
$ cd trafficserver
$ autoreconf -if
$ ./configure --prefix=/PATH/TO/ATS --with-quiche=/PATH/TO/QUICHE --enable-debug
$ make
$ make install
```

Build ATS (master branch)

The master branch only supports ATS's native implementation at the moment. There is no additional requirement except that you need the SSL library you just built : Installing From Source Code

```
$ git clone --depth 1 --branch quic-latest https://github.com/apache/trafficserver
$ cd trafficserver
$ autoreconf -if
$ ./configure --prefix=/PATH/TO/ATS --with-openssl=/PATH/TO/SSL_LIBRARY --enable-debug
$ make
$ make install
```

Configuration

Configuration files are located in the /PATH/TO/THE/ATS/etc/trafficserver/.

The detail is documented here, but below is the essential settings and only these 4 settings are available if you use Quiche.

- records.config

```
# run 1 UDP thread at least
CONFIG proxy.config.udp.threads INT 1

# open server port for quic
CONFIG proxy.config.http.server_ports STRING 4433:quic

# enable debug log if you want
CONFIG proxy.config.diags.debug.enabled INT 1
CONFIG proxy.config.diags.debug.tags STRING quic
```

```
> CONFIG proxy.config.udp.threads INT 1
23c24
< CONFIG proxy.config.http.server_ports STRING 8080 8080:ipv6
---
> CONFIG proxy.config.http.server_ports STRING 4433:quic
175,176c176,177
< CONFIG proxy.config.diags.debug.enabled INT 0
< CONFIG proxy.config.diags.debug.tags STRING http|dns
---
> CONFIG proxy.config.diags.debug.enabled INT 1
> CONFIG proxy.config.diags.debug.tags STRING quic
```

- ssl_multicert.config
  - Please use absolute path to the cert and private key until Issue #2358 is fixed.

```
dest_ip=* ssl_cert_name=/PATH/TO/THE/CERT ssl_key_name=/PATH/TO/THE/PRIVATE/KEY/OF/CERT
```

- remap.config
  - Remap request to origin server.

```
map / http://127.0.0.1:8000/
```

Run ATS

```
/PATH/TO/THE/ATS/bin/traffic_server
```

Patches

quic.ogre.com has additional patch to make debug logs readable.

QUIC specific configurations

Following docs will be moved to docs.trafficserver.apache.org.

Please note that current name of configurations and default values might be changed before merged in to master branch.

records.config

```
CONFIG proxy.config.quic.no_activity_timeout_in INT 30

   Specifies how long Traffic Server keeps QUIC connections to clients open if a transaction stalls.
```

How to test

Third-party tools

There is a script that builds third-party tools in the repo. It builds h2load and curl with HTTP/3 support. An HTTP/3 client under ngtcp2/example is also useful when you want to check details.

https://github.com/apache/trafficserver/blob/10-Dev/tools/build_h3_tools.sh

traffic_quic

We have client implementation called "traffic_quic" for test. Not actively maintained, and compatibility with Quiche implementation is not confirmed.

```
// draft-17

$ traffic_quic -h
Usage: traffic_quic [--SWITCH [ARG]]
  switch_____________________type__default___description
  -a, --addr               str   127.0.0.1 Address
  -o, --output             str             Write to FILE instead of stdout
  -p, --port               str   4433      Port
  -P, --path               str   /         Path
  -T, --debug              str   quic|vv.. Vertical-bar-separated Debug Tags
  -c, --close              on    false     Enable connection close excercise
  -h, --help                               Print usage information
  -V, --version                            Print version string
      --run-root                           using TS_RUNROOT as sandbox
```

client specific configurations

traffic_quic loads records.config which is used by traffic_server.

records.config

```
# Enable Version Negotiation Exercise
CONFIG proxy.config.quic.client.vn_exercise_enabled INT 1

# Enable Connection Migration Exercise
CONFIG proxy.config.quic.client.cm_exercise_enabled INT 1

# Enable TLS session resumption
CONFIG proxy.config.quic.client.session_file STRING session.bin
```

These configurations can be overridden by a corresponding environment variable like other configurations in records.config.

e.g. Access quic.ogre.com with version negotiation exercise

```
$ PROXY_CONFIG_QUIC_CLIENT_VN_EXERCISE_ENABLED=1 traffic_quic -a quic.ogre.com -p 4433 -P /en/latest/
```