QUIC Version Negotiation

draft-ietf-quic-version-negotiation

IETF 111 – Virtual San Francisco – 2021-07

David Schinazi – dschinazi@google.com
Eric Rescorla – ekr@rtfm.com

A brief history of QUIC Version Negotiation

2013: GoogleQUIC adds version negotiation and downgrade protection
2016-07: IETF QUIC initially had it too
2018-09: issues found with incremental server deployments Issue#1810
2019-02: removed VN from the base drafts to unblock them via PR#2313
2019-03: published draft-schinazi-quic-version-negotiation-00
2020-02: adopted as draft-ietf-quic-version-negotiation-00

2021-04: QUIC WG interim dedicated to version negotiation
  Consensus to keep compatible and incompatible but try to simplify

2021-05: draft-04 published with simplified design

RFC 9000 shipped, where does it stand on VN

Invariants (RFC 8999) define format of VN packet

RFC 9000 says that client aborts the connection on receipt of VN

Main use-case of HTTP/3 can survive without VN because of Alt-Svc

But QUIC is general-purpose

Requirement: allow VN without spending a round trip for similar versions

Incompatible Version Negotiation

Client sends first flight using version A ------->

<------- Server sends VN with list of support versions

Client sends another first flight with version B ------->

Compatible Version Negotiation

Client sends first flight using version A (listing compatible versions) ------->

<------- Server sends first flight with a compatible version

But what is a "Compatible Version" anyway?

Conceptually means that you can convert a first flight from one version to another

Note: not bijective, A compatible with B doesn't imply B compatible with A

Note: not all first flights need to be compatible (e.g., new frame added but not used)
client might profile its first flight to facilitate compatibility

Handshake Version Information

Sent during handshake – in QUICv1, uses transport parameter

Prevents downgrade attacks
Server performs verification to allow gradual deployment and multi-CDN

Allows exchanging compatible and supported versions

```
Version Information {
  Chosen Version (32),
  Other Versions (32) ...,
}
```

For client, Other Versions = Compatible Versions
For server, Other Versions = Supported Versions (some caveats apply)

Backup slide: draft-03 Handshake Version Information

```
Client Handshake Version Information {
   Currently Attempted Version (32),
   Previously Attempted Version (32),
   Received Negotiation Version Count (i),
   Received Negotiation Version (32) ...,
   Compatible Version Count (i),
   Compatible Version (32) ...,
}
```

```
Server Handshake Version Information {
   Negotiated Version (32),
   Supported Version Count (i),
   Supported Version (32) ...,
}
```

Server Supported Versions

In a simple server deployment, the server can send the same set of versions in VN packets and in Handshake Version Information

This can cause connection failures during versions additions or removals

Section 2 describes a three-step algorithm for progressively adding or removing versions without failures, where these sets are updated progressively:

- Acceptable Versions: versions the server will speak if asked
- Offered Versions: versions the server sends in VN packets
- Fully-Deployed Versions: versions the server sends in HVI Other Versions

Algorithm adds complexity but is a SHOULD as some servers don't care

Downgrade Protection

Clients ignore VN that contains the attempted version

If client has reacted to VN, it has used a version selection algorithm to pick new version from the set of Supported Versions in VN pakcet

Client validates server's Other Versions by re-running the version selection algorithm on the set offered in Other Versions, aborts if different result

Issue #43: Client version preference

Currently the client's "Other Versions" field (which lists compatible versions) is ordered by preference.

This could be useful but isn't strictly required.

Opinions?

QUIC Version Negotiation

draft-ietf-quic-version-negotiation

IETF 111 – Virtual San Francisco – 2021-07

David Schinazi – dschinazi@google.com
Eric Rescorla – ekr@rtfm.com