|  |  |
| --- | --- |
| 3GPP TR 29.893 V18.0.0 (2023-09) | |
| Technical Report | |
| 3rd Generation Partnership Project;  Technical Specification Group Core Network and Terminals;  Study on IETF QUIC Transport for 5GC Service Based Interfaces;  (Release 18) | |
|  | |
|  |  |
|  | |
| The present document has been developed within the 3rd Generation Partnership Project (3GPP TM) and may be further elaborated for the purposes of 3GPP. The present document has not been subject to any approval process by the 3GPPOrganizational Partners and shall not be implemented. This Specification is provided for future development work within 3GPPonly. The Organizational Partners accept no liability for any use of this Specification. Specifications and Reports for implementation of the 3GPP TM system should be obtained via the 3GPP Organizational Partners' Publications Offices. | |

|  |
| --- |
|  |
| ***3GPP***  Postal address  3GPP support office address  650 Route des Lucioles - Sophia Antipolis  Valbonne - FRANCE  Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16  Internet  http://www.3gpp.org |
| ***Copyright Notification***  No part may be reproduced except as authorized by written permission. The copyright and the foregoing restriction extend to reproduction in all media.  © 2023, 3GPP Organizational Partners (ARIB, ATIS, CCSA, ETSI, TSDSI, TTA, TTC).  All rights reserved.  UMTS™ is a Trade Mark of ETSI registered for the benefit of its members  3GPP™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners LTE™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners  GSM® and the GSM logo are registered and owned by the GSM Association |

Contents

Foreword 5

1 Scope 6

2 References 6

3 Definitions, symbols and abbreviations 8

3.1 Definitions 8

3.2 Abbreviations 8

4 Architectural Baseline 8

5 Transport Protocol Features Required For 3GPP 5GC SBI 9

5.1 Introduction 9

5.2 Requirements from Transport Protocol for 3GPP 5GC SBI 9

5.3 Features of QUIC 10

5.3.1 General 10

5.3.2 Framing and Multiplexing 10

5.3.3 Improved Recovery and Acknowledgement 11

5.3.4 Encrypted and Integrity Protected Transport details 11

5.3.5 Connection Setup Improvements 11

5.3.6 0-RTT Data 12

5.3.7 Connection ID 12

5.3.8 Connection Migration 12

5.3.9 Stream Prioritization 13

5.3.10 Flow Control 13

5.3.11 Protocol Versioning 13

5.3.12 QUIC Extensibility 13

5.3.13 Connection Configuration 14

5.3.14 User-Land Implementations 14

5.3.15 Pluggable Sender Side Congestion Control 14

5.3.16 Checking that the QUIC connection is alive 14

5.3.17 62 bits stream identifiers 14

5.3.18 Running atop of UDP 15

5.4 Features of QUIC Applicable to 3GPP SBI 15

5.4.1 General 15

5.4.2 Framing and Multiplexing 15

5.4.3 Encrypted and Integrity Protected Transport details 15

5.4.4 Connection setup improvements 15

5.4.5 Connection ID and Connection Migration 16

5.4.6 Improved Recovery and Acknowledgement 16

5.5 Features of QUIC Not Applicable to 3GPP SBI 16

5.5.1 General 16

5.5.2 0-RTT DATA 16

5.6 Comparison of Applicable Features with Rel-15 Transport 17

6 HTTP/3 20

6.1 Introduction 20

6.2 HTTP/3 Proxies 21

6.2.1 General 21

6.2.2 When NF Service Consumer Side Uses QUIC 22

6.2.2.1 Case A: Invoking http API Supporting Only TCP Transport 22

6.2.2.2 Case B: Invoking http API Supporting QUIC Transport 24

6.2.2.3 Case C: Invoking https API Supporting Only TCP Transport 25

6.2.2.4 Case D: Invoking https API Supporting QUIC Transport 26

6.2.3 When NF Service Consumer Side Uses TCP 27

6.2.3.1 Invoking http API Supporting QUIC Transport 27

6.2.3.2 Invoking https API Supporting QUIC Transport 27

6.3 Considerations for HTTP/3 28

6.3.1 General 28

6.3.2 Connection setup and management 28

6.3.3 Streams, framing and multiplexing 28

6.3.4 Prioritization 29

6.3.5 Server Push 29

6.3.6 Compression (HPACK vs QPACK) 29

7 Key Requirements for Supporting QUIC 29

7.1 Introduction 29

7.2 Discovery of QUIC support 29

7.3 Discovery of NRF's Support for QUIC 30

7.4 Migration to QUIC 30

7.5 Support of Indirect Communication 30

8 Solutions for Key Requirements 30

8.1 Introduction 30

8.2 Solutions for Discovery of QUIC support 31

8.2.1 Using the Discovery Service of the NRF 31

8.2.2 Using Alt-Svc Header 31

8.3 Solutions for Discovery of NRF's Support for QUIC 32

8.3.1 Providing NRF Transport Capability from NSSF 32

8.3.2 Providing Remote PLMN NRF's Transport Capability during NF Discovery 32

8.3.3 Discovery Based On Local Configuration 32

8.4 Solutions for Migration to QUIC 33

8.4.1 Deployment Topologies to Introduce NF Services with QUIC Support 33

8.4.2 Steps to Follow When Introducing NF Services with QUIC Support 33

8.4.3 Use of QUIC by NF Service Consumers 33

8.4.4 Decommissioning TCP 33

9 Impacts to Service Based Architecture 34

9.1 Introduction 34

9.2 HTTP Proxy Traversal 34

9.3 QUIC's Security Mechanisms 34

9.4 TCP Decommission in Migration Impacts Architecture 34

9.5 Transport Proxy Traversal 35

9.6 Impacts on Troubleshooting 35

9.6.1 Introduction 35

9.6.2 QUIC keying impact 35

9.6.3 Network level troubleshooting 35

9.6.4 Application level troubleshooting 36

10 Current Implementation and Maturity Status 36

10.1 Introduction 36

10.2 Implementation maturity 36

10.3 Hardware offload support 37

10.4 UDP Performance in Operating System 37

11 Evaluation and Conclusion 38

11.1 Evaluation 38

11.1.1 Requirements for Service Based Interfaces 38

11.1.2 Expected improvements 38

11.1.3 Issues 38

11.1.4 Other considerations 39

11.2 Conclusion 39

Annex A (informative): HTTP/2 vs. HTTP/3 testing findings and considerations 39

Annex B: Change history 41

# Foreword

This Technical Report has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the document.

In the present document, modal verbs have the following meanings:

**shall** indicates a mandatory requirement to do something

**shall not** indicates an interdiction (prohibition) to do something

The constructions "shall" and "shall not" are confined to the context of normative provisions, and do not appear in Technical Reports.

The constructions "must" and "must not" are not used as substitutes for "shall" and "shall not". Their use is avoided insofar as possible, and they are not used in a normative context except in a direct citation from an external, referenced, non-3GPP document, or so as to maintain continuity of style when extending or modifying the provisions of such a referenced document.

**should** indicates a recommendation to do something

**should not** indicates a recommendation not to do something

**may** indicates permission to do something

**need not** indicates permission not to do something

The construction "may not" is ambiguous and is not used in normative elements. The unambiguous constructions "might not" or "shall not" are used instead, depending upon the meaning intended.

**can** indicates that something is possible

**cannot** indicates that something is impossible

The constructions "can" and "cannot" are not substitutes for "may" and "need not".

**will** indicates that something is certain or expected to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document

**will not** indicates that something is certain or expected not to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document

**might** indicates a likelihood that something will happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document

**might not** indicates a likelihood that something will not happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document

In addition:

**is** (or any other verb in the indicative mood) indicates a statement of fact

**is not** (or any other negative verb in the indicative mood) indicates a statement of fact

The constructions "is" and "is not" do not indicate requirements.

# 1 Scope

The present document analyses the IETF QUIC protocol and its potential use as a transport protocol for the 5GC Service Based Interfaces.

This technical report provides an analysis of the following aspects:

- Features of transport protocol required for 5GC SBI;

- Features of QUIC applicable to 5GC SBI;

- Comparison of the applicable features of QUIC against TCP for the 5GC SBI;

- Key requirements for 5GC SBI in order to support QUIC as a transport protocol;

- Solutions for supporting the key requirements;

- Impacts to 5GC Service Based Interfaces due to introduction of QUIC.

# 2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non‑specific.

- For a specific reference, subsequent revisions do not apply.

- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 23.501: "System Architecture for the 5G System; Stage 2".

[3] 3GPP TS 23.502: "Procedures for the 5G System; Stage 2".

[4] 3GPP TS 29.500: "5G System; Technical Realization of Service Based Architecture; Stage 3".

[5] IETF RFC 9000: "QUIC: A UDP-Based Multiplexed and Secure Transport".

[6] IETF RFC 9001: "Using TLS to Secure QUIC".

[7] IETF RFC 9114: "HTTP/3".

[8] IETF RFC 9002: "QUIC Loss Detection and Congestion Control".

[9] IETF RFC 8999: "Version-Independent Properties of QUIC".

[10] IETF RFC 9204: "QPACK: Header Compression for HTTP/3"

[11] IETF RFC 5246: "The Transport Layer Security (TLS) Protocol Version 1.2".

[12] IETF RFC 8446: "The Transport Layer Security (TLS) Protocol Version 1.3".

[13] IETF RFC 7540: "Hypertext Transfer Protocol Version 2 (HTTP/2)".

[14] IETF RFC 7541: "HPACK: Header Compression for HTTP/2".

[15] Void

[16] IETF RFC 5682: "Forward RTO-Recovery (F-RTO): An Algorithm for Detecting Spurious Retransmission Timeouts with TCP".

[17] IETF draft-dukkipati-tcpm-tcp-loss-probe-01: "Tail Loss Probe (TLP): An Algorithm for Fast Recovery of Tail Losses".

[18] IETF RFC 6582: "The NewReno Modification to TCP's Fast Recovery Algorithm".

[19] 3GPP TS 29.510: "Network Function Repository Services".

[20] IETF RFC 7838: "HTTP Alternative Services".

[21] IETF draft-pardue-httpbis-http-network-tunnelling-01: "HTTP-initiated Network Tunnelling (HiNT)".

[22] IETF RFC 7231: "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content".

[23] IETF RFC 7230: "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing".

[24] 3GPP TS 33.210: "3G security; Network Domain Security (NDS); IP network layer security".

[25] GSMA NG.113: "5GS Roaming Guidelines".

[26] IETF RFC 8312: "CUBIC for Fast Long-Distance Networks".

[27] 3GPP TR 23.742: "Study on Enhancements to the Service-Based Architecture".

[28] IETF RFC 8164: "Opportunistic Security for HTTP/2".

[29] IETF RFC 7657: "Differentiated Services (Diffserv) and Real-Time Communication".

[30] Taking a Long Look at QUIC: "https://conferences.sigcomm.org/imc/2017/papers/imc17-final39.pdf".

[31] IETF RFC 5288: "AES Galois Counter Mode (GCM) Cipher Suits for TLS".

[32] Developing and deploying a TCP replacement for the Web: "https://www.netdevconf.org/0x12/session.html?developing-and-deploying-a-tcp-replacement-for-the-web".

[33] Optimizing UDP for content delivery: "http://vger.kernel.org/lpc\_net2018\_talks/willemdebruijn-lpc2018-udpgso-paper-DRAFT-1.pdf".

[34] UDP segmentation offload: "https://www.netdevconf.org/0x12/session.html?udp-segmentation-offload".

[35] IETF draft-ietf-masque-connect-ip-13: "Proxying IP in HTTP".

[36] IETF RFC 9298: "Proxying UDP in HTTP".

[37] IETF draft-ietf-quic-multipath-04: "Multipath Extension for QUIC".

[38] QUIC vs TCP: Which is Better?: "https://www.fastly.com/blog/measuring-quic-vs-tcp-computational-efficiency".

[39] IETF draft-ietf-quic-ack-frequency-04: "QUIC Acknowledgement Frequency", "https://datatracker.ietf.org/doc/draft-ietf-quic-ack-frequency".

[40] Comparison of Different QUIC Implementations: "https://www.net.in.tum.de/fileadmin/TUM/NET/NET-2022-07-1/NET-2022-07-1\_10.pdf".

[41] How Facebook is bringing QUIC to billions: "https://engineering.fb.com/networking-traffic/how-facebook-is-bringing-quic-to-billions".

[42] Comparing HTTP/3 vs. HTTP/2 Performance: "https://blog.cloudflare.com/http-3-vs-http-2".

[43] H2O the optimized HTTP/1.x, HTTP/2 server: "https://h2o.examp1e.net/benchmarks.html".

[44] Of the Utmost Importance: Resource Prioritization in HTTP/3 over QUIC: "https://h3.edm.uhasselt.be/files/HTTP3\_Prioritization\_extended\_3jul2019.pdf".

[45] Head-of-Line Blocking in QUIC and HTTP/3: The Details: "[https://calendar.perfplanet.com/2020/head-of-line-blocking-in-quic-and-http-3-the-details](https://calendar.perfplanet.com/2020/head-of-line-blocking-in-quic-and-http-3-the-details/)".

# 3 Definitions, symbols and abbreviations

## 3.1 Definitions

For the purposes of the present document, the terms and definitions given in 3GPP TR 21.905 [1] and the following apply. A term defined in the present document takes precedence over the definition of the same term, if any, in 3GPP TR 21.905 [1].

## 3.2 Abbreviations

For the purposes of the present document, the abbreviations given in 3GPP TR 21.905 [1] and the following apply. An abbreviation defined in the present document takes precedence over the definition of the same abbreviation, if any, in 3GPP TR 21.905 [1].

RTT Round Trip Time

TLS Transport Layer Security

# 4 Architectural Baseline

3GPP Release 16 Service Based Architecture as specified in 3GPP TS 23.501 [2] and the 3GPP Release 16 Technical Realization of the Service Based Architecture as specified in 3GPP TS 29.500 [4] shall be taken as the baseline for studying QUIC's use as a transport protocol for the 5GS Service Based Interfaces.

In particular the following architectural assumptions shall be taken into account:

- Replacing the transport protocol from TCP to QUIC shall not lead to any change in the semantics of the NF services and shall not lead to any change in API.

- SEPP shall be used as the security protection and edge proxy even when the NF service consumer in VPLMN and the NF service consumer in HPLMN both use QUIC as the transport.

- For inter PLMN NF service communication, even if both the NF service consumer and NF service producer support QUIC, the IPX providers and intermediaries on path between the two NF's first hop and the last hop shall not be mandated to support QUIC. In other words, the NF service consumer and the NF service producer shall be able to communicate when using QUIC as transport even in the presence of TCP based IPX or intermediaries on path between the first hop and the last hop.

- A NF Service Consumer and NF Service Producer may communicate indirectly via Service Communication Proxy (SCP). The indirect communication options defined in Annex E of 3GPP TS 23.501 [2]) and listed hereafter shall be permitted:

- Option C - Indirect communication without delegated discovery;

- Option D - Indirect communication with delegated discovery.

SCPs act as HTTP proxies. SCPs require the ability to look into the HTTP message for delegated discovery and selection functionality.

# 5 Transport Protocol Features Required For 3GPP 5GC SBI

## 5.1 Introduction

This clause will identify the requirements from transport protocol for the 3GPP 5GC service based interfaces and then subsequently highlights the features of the QUIC protocol. Among the features of QUIC, those features that are applicable for the 3GPP 5GC service based interfaces are then explicitly identified.

## 5.2 Requirements from Transport Protocol for 3GPP 5GC SBI

The following are the key requirements from transport layer protocols for the 3GPP 5GC service based interfaces. The 3GPP 5GC service based interfaces use HTTP/2 as the application layer protocol. As of 3GPP Release 15, TCP is used as the transport protocol for the 5GC service based interfaces. TCP suffers from Head of Line blocking issues and any new transport protocol that is considered as a replacement to TCP for the 3GPP 5GC service based interfaces shall not have the same limitation.

REQ#1: The transport layer protocol shall support reliable message delivery.

REQ#2: The transport layer protocol shall support flow control and congestion control mechanisms.

REQ#3: The transport layer protocol shall support connection semantics as required by IETF RFC 7540 [13].

REQ#4: The failure to deliver one message shall not block subsequent messages.

REQ#5: The transport layer protocol shall have mechanisms to allow authentication of the peer transport endpoint and shall have mechanisms for the secure transfer of application layer messages.

REQ#6: The transport layer protocol shall have mechanisms to allow processing of the HTTP/2 messages over it by intermediaries (e.g. proxies like SCP).

REQ#7: The transport layer protocol should support troubleshooting and Monitoring, e.g. message trace and parse in the middlebox for the scenarios of testing, monitoring, troubleshooting and etc.

REQ#8: The transport layer protocol, along with the upper layer application protocol shall support the distributed architecture for PNFs and VNFs, e.g. the front-end load-balancer and back-end service process-units architecture.

REQ#9: The performance and resource efficiency (i.e. CPU, Memory and other processing requirements) shall be one of the evaluation criteria of the transport layer protocol.

REQ#10: The complexity and cost of development and deployment of the transport layer protocol shall be one of the evaluation criteria.

REQ#11: The transport layer protocol shall be mature enough for adoption of the protocol in the 5GC, e.g. fully standardized and has mature open source support, not only for the QUIC protocol, but also the full stack of HTTP/3 protocols.

## 5.3 Features of QUIC

### 5.3.1 General

QUIC is a multiplexed and secure transport protocol that runs on top of UDP. QUIC aims to provide a flexible set of features that allow it to be a general-purpose secure transport for multiple applications. The main parts of QUIC are defined in a set of documents IETF RFC 9000 [5], IETF RFC 9002 [8], IETF RFC 9001 [6], IETF RFC 8999 [9]. The highly integrated HTTP/2 over QUIC specification (now called HTTP/3) IETF RFC 9114 [7] and HTTP header compression IETF RFC 9204 [10] are developed in parallel with the core protocol. The protocol is developed by the Internet Engineering Task Force (IETF).

QUIC is mainly designed for the communication across insecure and untrusted internet, it integrates some features to tackle the performance, security and privacy related challenges, the applicability of applying QUIC in 5G Core shall be evaluated.

### 5.3.2 Framing and Multiplexing

Provisions in this clause are based on IETF RFC 9000 [5]. QUIC endpoints communicate by exchanging QUIC packets in UDP datagrams. One or more QUIC packets can be encapsulated in a single UDP datagram. QUIC packets may have long or short headers, for packets sent prior or after the completion of version negotiation and establishment of 1-RTT keys respectively. A sender multiplexes one or more QUIC frames of the same or different type(s) into a QUIC packet. The header contains only a limited set of fields, including Destination Connection ID field (see clause 12) and encrypted Packet Number field. Data sent by an application is encapsulated in STREAM frames which are carried in 0-RTT packets, if send as early data during the handshake, or afterwards in 1-RTT packets. An endpoint uses the Stream ID and Offset fields in STREAM frames to place data in order. The usage of the Offset field is further explained by the below example.

EXAMPLE: Let's say a QUIC transport entity sends data via three STREAM frames. The first one is e.g. 10 octets long, starts at offset 0 and ends at offset 9. The second one is e.g. 20 octets long, starts at offset 10 and ends at offset 29. The third one is e.g. 7 octets long, starts at offset 30 and ends at offset 36. Let's also assume these frames are carried by separate QUIC packets A, B and C. In a sunny day scenario all three packets arrive at the receiving entity, which successfully acknowledges them. In a rainy day scenario, retransmissions are necessary. Let's say, packets A and B were lost and only packet C is delivered and acknowledged. The sender obviously needs to retransmit the content with offset 0..9 and 10..29. The sender can do this either by resending these two frames as initially sent (i.e. the three frames) or by sending a single, 10+20=30 octets long frame with offset 0..29.

This is one of the differences with TCP, as TCP only provides one stream and all data therefore are delivered in order, which means multiplexing is not supported in TCP. A sender can wait for a short period of time to bundle multiple frames into the same QUIC packet, e.g. to minimize the computational costs of packets sending.

QUIC supports multiple parallel data streams multiplexed on a single QUIC connection. Streams, which can be unidirectional or bidirectional in QUIC provide a lightweight, ordered byte-stream abstraction to an application. Packets transmitted in each stream use Authenticated Encryption with Additional Data (AEAD) to provide confidentiality and integrity protection. Streams can be long-lived, even during the lifetime of a connection to increase the reusability and limit the cost of opening stream (See IETF RFC 9000 [5]). An endpoint of a bidirectional stream can terminate one direction and even encourage prompt termination in the opposite direction.

For each stream QUIC now only supports reliable and in-order delivery, but the implementations may choose to offer the ability to deliver data out of order. However, the QUIC layer is capable of delivering to the higher layer each stream independently as the streams in QUIC are independent of each other, thus it avoids blocking the delivery of any of the other streams when a packet loss contains only part of a stream which would be the case for HTTP/2 over TCP. Note that to achieve this efficiency the implementation needs to pay attention to pack payload from one stream into a single QUIC packet.

The HTTP/3 mapping for QUIC IETF RFC 9114 [7] utilizes this stream concept when realizing the different HTTP/2 (See IETF RFC 7540 [13]) streams. HTTP/3 also had to improve the HTTP header compression scheme HPACK (See IETF RFC 7541 [14] into QPACK (See IETF RFC 9204 [10]). With these changes HTTP can deliver independent requests and responses in the order they are successfully delivered to endpoints, without head of line blocking between HTTP streams which would be the case for HTTP/2 over TCP.

### 5.3.3 Improved Recovery and Acknowledgement

The QUIC definition of its packet format and acknowledgement frame results in several improvements over TCP. The packet number is transmission-time ordered and strictly increasing. QUIC never retransmits a particular packet, only the lost data frames that need to be retransmitted. QUIC facilitates better way to calculate RTT by encoding the delay between packet reception and transmission of the acknowledgement. The QUIC acknowledgment also supports a very larger number of received and gap ranges.

Compared to TCP, QUIC will not be limited to a three blocks of selective acknowledgement (SACK) when using the timestamp option. Each ACK Frame in QUIC can contain variable number of ACK ranges, up to 62 bits (See IETF RFC 9000 [5]), which helps to ease network throughputs in case of sending packets frequently. The strict packet numbers and explicit acknowledgement removes ambiguity between which packet is lost and which is acknowledged. Avoiding any unnecessary retransmissions of data that have reached the receiver. QUIC also avoids the retransmission uncertainty if the received packet was a delayed or retransmitted. QUIC's RTT samples are more accurate than what TCP can provide due to no ambiguity about which packets are used in measurement as well as the receiver side delay can be taken into account.

The congestion control algorithm of the current QUIC version is based on NewReno (See IETF RFC 6582 [18]), but implementations can use other congestion control algorithms, such as Cubic (see IETF RFC 8312 [26]), and endpoints are allowed to use different algorithms from one another. QUIC can customise different congestion control algorithms for different connections of the same application, and even alter it during the lifetime of a connection, see clause 9.4. QUIC provides generic congestion control signals to support different algorithms. QUIC also uses some additional modern loss recovery mechanisms by default, such as F-RTO (See IETF RFC 5682 [16]), and Tail Loss Probing (See IETF draft-dukkipati-tcpm-tcp-loss-probe [17]). These improvements give QUIC a better recovery mechanism.

### 5.3.4 Encrypted and Integrity Protected Transport details

QUIC uses TLS 1.3 (See IETF RFC 9001 [6], IETF RFC 8446 [12]), for key establishment, QUIC integrates the TLS 1.3 as its own encryption and integrity layer that protects the QUIC packets, but the security capability of HTTP/3 over QUIC/UDP is consistent with HTTP/2 over TLS1.3/TCP. Each QUIC packet has a packet header, using a short or a long format with a small number of fields that are unencrypted, but integrity protected. It is primarily the connection ID that is unencrypted and three reserved bits for experimentation in the short header. Even the packet number is encrypted using an independent mechanism from the payload.

The encryption and integrity protection provide confidentiality, privacy and source authenticity for the user of QUIC. However, the protection is also intended to prevent any middlebox in the network from interfering with the protocol, nor make assumptions about what the possible values any specific bit in the UDP payload can take. Ossification of the network has prevented a lot of improvements from being applied to TCP as middleboxes would either block or remove such changes.

Compared to TCP, this level of encryption does make certain type of network performance monitoring using middlebox basically impossible. Due to this, the QUIC short header introduces a latency spin bit (See IETF RFC 9000 [5]) that is intended to enable middlebox to measure round-trip time between the middlebox and either endpoint of the connection if enabled by both end-points. The latency spin bit partially overcomes the drawback of impossible network performance monitoring caused by encryption in QUIC layer, but cannot support the message tracing by content inspection for the testing, monitoring and troubleshooting related scenarios.

### 5.3.5 Connection Setup Improvements

QUIC is capable of completing establishment of a connection between a client and a server in one and half RTT. The protocol combines TLS (See IETF RFC 8446 [12]) handshake with transport protocol level mechanisms to achieve this. A client's request to a server can be included after one RTT and be sent combined with the last step of the crypto handshake from the client to the server.

Holding state in the server for the initial connection establishment packets prior to having verified the client's return path can expose the server to a denial of service risk. Servers that like to mitigate that risk can use the Retry packet to verify the path and not hold any state for the first round-trip.

How big improvement this is depends on what one compares against. As 3GPP TS 33.310 makes support for TLS 1.3 (See IETF RFC 8446 [12]) mandated from Rel-15 it is reasonable to compare with both TLS 1.2 (See IETF RFC 5246 [11]) and TLS 1.3. TLS 1.2 session resumption requires that the client has talked to the server recently enough, so it still has session state stored. The below table indicates number of RTTs until the first HTTP request can be sent by the client.

Table 5.3.5-1: Number of RTTs until first HTTP request

|  |  |  |
| --- | --- | --- |
| Protocol | New Connection | Connection State Exists |
| TCP/TLS 1.2 | 3 | 2 |
| TCP/TLS 1.3 | 2 | 1 |
| QUIC | 1 | 0 |

QUIC can achieve faster connection establishment times until an HTTP request has been sent than existing TLS and TCP combinations. This improvement is significant when establishing a new connection, but not when clients have a long lived one to the server.

### 5.3.6 0-RTT Data

TLS 1.3 (IETF RFC 8446 [12]) includes support for early data or 0-RTT data, as it is also called. This is potentially usable by both HTTP/2 over TLS1.3/TCP as well as HTTP/3. This functionality can be used when client and server share a Pre-Shared Key (PSK), which can be arranged out of band or exist from an earlier connection. 0-RTT data has other security properties than for data sent after the handshake completes. Data sent as 0-RTT data will be possible to replay by an attacker that has seen the client to server exchange. Therefore, the use of 0-RTT data requires that the data is safe to replay. When using HTTP requests as 0-RTT data, the request performed must be one that is idempotent. Server may refuse to accept 0-RTT data for this reason.

A server accepts 0-RTT data on a connection needs more processing and computation cost. Servers need to consider the probability of replay and all associated costs when accepting 0-RTT (See IETF RFC 9001 [6], IETF RFC 8446 [12]).

### 5.3.7 Connection ID

QUIC uses a set of connection IDs, one connection ID for the server and one connection ID for the client to identify a particular connection for an endpoint. During the handshake, QUIC packet with the long header is used to exchange the connection ID that each endpoint assigned. The endpoint is allowed to change the own connection ID to another available one at any time during the connection without any interruption in the transmission. This solution makes the connection not hard bound to a particular 5-tuple (Source and Destination IP, protocol, and source and destination port), instead the connection can be moved between different network interfaces on client and with some limitations on the server side. The protocol has a feature for migrating connections from using one 5-tuple to another, see clause 5.3.8. When knowingly changing the used 5-tuple a new connection ID is necessarily to be used. The peers exchange additional connection IDs when needed to ensure that the peer have one or more previously unused CIDs that can be used in case of connection migration. The middlebox is difficult to correlate the received packet to the connection as the procedure used to changing connection is in encryption.

The length of connection ID is variable, and it provides certain flexibility in how the implementers realize network equipment architecture, e.g. front-end load-balancers, for QUIC.

### 5.3.8 Connection Migration

QUIC allows its connection to migrate while the HTTP/3 session progresses. This means for a client with multiple network interfaces an ongoing QUIC session can be moved to newly validated path via a newly discovered network interface, for example, in the case of a data session handover from WLAN to a 3GPP radio access technology. This is possible as QUIC sessions are identified by a set of connection IDs hence a particular QUIC session is not tightly coupled with a specific client IP address and port number. If a network interface appears with new IP addresses or an existing one disappears but the client has alternative network interfaces, the QUIC session does not need to be established again. The QUIC session can continue on a new interface after the client has validated the path to the server from the new interface address using PATH\_CHALLENGE frames, with potentially a new connection ID from the previously communicated set of connection IDs.

NOTE: IETF RFC 9000 [5] does not mandate a new connection ID after connection migration. However such reuse is not recommended as this allows on path observers to link multiple source IP addresses to the same connection and identify the topological relationship of clients. See clause 9.5 of IETF RFC 9000 [5].

It is possible that the server also has multiple IP addresses and has some preferences on which interface it would like to serve a particular client for load balancing or other management. QUIC allows server to receive a connection request to one IP address and migrate the connection to a preferred address in connection response immediately, this achieves faster connection migration than HTTP redirect. Currently, QUIC does not support change of server IP address in the middle of an ongoing session however, the server preferred address can be conveyed to the client during the TLS handshake as "preferred\_address" transport parameter (see clause 9.6 of IETF RFC 9000 [5]). If the new path to the preferred server address is valid then client sends all the future packets to the new server address. Here the client also uses a new connection ID for the new connection to the server's preferred address.

### 5.3.9 Stream Prioritization

Being a multiplexed transport protocol, QUIC supports stream prioritization for boosting the application performance. However, QUIC itself neither provides mechanism to negotiate prioritization information nor implements any strict prioritization scheme. It relies on the application to provide priority information that QUIC will follow when it comes to packet transmission or retransmission. HTTP/3 uses the same prioritization mechanism as HTTP/2.

### 5.3.10 Flow Control

Flow control is a mechanism to set boundaries to the senders to avoid overwhelming receiver with data that the receiver cannot process. Like TCP, QUIC deploys connection level flow control, moreover, it applies a secondary stream level flow control to prevent a particular stream from consuming the receiver buffer for a connection. As QUIC provides multiplexing, flow control from the HTTP/2 layer is moved to the QUIC layer, avoiding head of line blocking and simplifying HTTP/3.

### 5.3.11 Protocol Versioning

QUIC has a 32-bit version field. It can be expected that QUIC will eventually exists in a number of proprietary and standardized versions. IETF is currently working on defining version 0x00000001. There exists a mechanism for the client to ask the server to enumerate all versions it support. The client when requesting to create a connection it will indicate the version desired to use. If supported then that is what will be used, otherwise it triggers the version negotiation. Some of the non-encrypted fields are defined as not being changeable independent of version as defined by the document for invariants (See IETF RFC 8999 [9]).

The QUIC versioning enables a very large degree of flexibility for future changes of QUIC. All aspects except for the invariants can be changed. This enables the tuning of QUIC to a specific use case or implementation of future improvements in transport protocol technology. This flexibility also indicates the need to be explicit about which QUIC version(s) that are to be supported by a specific SBI. Any analysis of benefits and downsides of QUIC must be explicit about which version is discussed.

### 5.3.12 QUIC Extensibility

QUIC payloads are consists of one or more frames. Each frame starts with frame types followed by type specific flags. All the streams with data are carried over the STREAM frame type. QUIC's current specification defines a number of essential frame types. However, new frame types can be created and can be even application specific.

QUIC allows extensions to the protocol within the constraints of the protocol invariants (see properties of the QUIC transport protocol that are expected to remain unchanged as new versions of the protocol are developed, in IETF RFC 8999 [9]). Extensions can change the semantics of existing protocol components, but they need to be negotiated before being used. Permitted extensions include new frame types, new settings, error codes and uni-directional streams. This gives QUIC a unique way of to be extensible and customizable.

The usage of new frame types does not necessarily imply using a new protocol version. A peer can use transport parameters to indicate support to the peer that it can use a new frame type. However, this has the downside that the support of a certain frame type cannot be determined before establishing the transport connection; on the other hand, using a specific protocol version can be leveraged by a peer to determine this support prior to establishing the connection.

### 5.3.13 Connection Configuration

QUIC allows a connection to be configured in a particular way with a set of transport parameters and frames. The transport parameters are exchanged in the cryptographic handshake. QUIC frames are used to configure how endpoints communicate. For example - the PADDING frame allow to vary the packet size, MAX\_STREAM\_ID frame indicates the maximum bidirectional or unidirectional stream ID permitted to open for the connection. Moreover, new transport parameters and frames can be added to extend the configuration.

### 5.3.14 User-Land Implementations

User space implementations of QUIC do not require elevated permissions. This allows application to include a QUIC implementation without any operating system changes. This simplifies deployment of QUIC, where only the application intending to use QUIC needs to be updated. This flexibility can also be used to fine tune the protocol behaviour to a particular application. However, there exists some risk with this, as even if an implementation is following the requirement of a certain QUIC version, the choices to optimize the implementation may result in poorer performance between two differently optimized implementations. Running separate QUIC applications in each application's user space has the impact on performance and resource efficiency in large-scale deployment scenario, which may not be a big problem for equipment with specific function in 5G Core.

The implementation in user space also results in certain challenges that can affect performance. Efficiency of the API towards the UDP receive and send functions is one such case. Other complications can be access to high performance timers and operating system's scheduling granularity.

### 5.3.15 Pluggable Sender Side Congestion Control

As QUIC implementation can reside in an application, it allows more experiment with congestion control algorithms. QUIC can customise different congestion control algorithms for connections of the same application. Compare to TCP, pluggable sender side congestion control makes it pretty flexible and effective on updating or terminating congestion control algorithms without upgrading operation system. However, the fairness of the bandwidth competition within the same network among applications using different congestion control algorithms needs to be considered.

Now depending on the operational environment, network and service requirement very specific congestion control algorithm can be deployed in the sender as long as the information in the acknowledgement from receiver is sufficient.

### 5.3.16 Checking that the QUIC connection is alive

Another difference with HTTP/2 is that each QUIC endpoint declares an idle timeout during the handshake. If the connection remains idle (no packets received) for longer than the advertised idle timeout, the peer will assume that the connection has been closed.

HTTP clients are expected to use QUIC PING frames to keep connections open if necessary, to verify that their peers are still alive or to check reachability to the peer. Without using QUIC PING frames, an inactive connection will time out. The frequency of sending PING frames is controlled by applications.

### 5.3.17 62 bits stream identifiers

QUIC stream identifiers are coded as variable length integers allowing upto a length of 62 bits, instead of 31 bits with HTTP/2. Out of the available 62 bits for stream ID encoding, 2 least significant bits are used to indicate who initiates the stream (client / server) and whether the stream is unidirectional or bidirectional. Hence for client initiated bidirectional streams to carry the requests and responses of 3GPP Service Based Interfaces, the available space is 2^60 stream IDs.

NOTE: For variable length integer encoding the 2 MSB bits are used to derive the length of the integer. The 2 MSB bits are coded as base 2 logarithm of the total length of the variable length integer in octets. Thus for a 64 bit sized entity, the 2 MSB bits are 11 (i.e. value 3), indicating that the length of the integer is 8 octets of which only 62 bits are usable. See clause 16 and clause 2.1 of IETF RFC 9000 [5].

Stream ID exhaustion becomes nearly impossible during the lifetime of a QUIC connection. This may simplify the management of connections in 5GC.

### 5.3.18 Running atop of UDP

When comparing HTTP/3 with HTTP/2, one should compare the complete protocol stack. HTTP/2 runs either on top of TCP or on top of TLS/TCP, while HTTP/3 runs on top of QUIC/UDP, where QUIC natively incorporates TLS. In other words, the difference goes well beyond replacing TCP with UDP, as TCP is mainly replaced by QUIC/UDP from the transport layer point of view. The purpose for using UDP in HTTP/3 stack is to ensure that QUIC packets pass transparently through legacy middleboxes including OS, router, firewall etc. However, due to DDoS attack avoidance, or other network operating consideration, network operators configure the network to limit the peak rate of UDP packets, which will heavily impact the exact performance behavior of QUIC and delay the popularity of QUIC deployment in particular districts. It could be a kind of deadlock for QUIC traffic used in internet until HTTP/3 is wildly accepted. For SBI being used within Telco network, this will not be a big issue.

## 5.4 Features of QUIC Applicable to 3GPP SBI

### 5.4.1 General

This clause reviews the features of HTTP/3 and QUIC that are applicable to 3GPP SBI and under which cases and conditions they are applicable.

### 5.4.2 Framing and Multiplexing

This feature allows QUIC to multiplex multiple streams in to a single connection and avoid head of line blocking. The upper layer protocols can use the QUIC transport in efficient ways to prioritize, parallelize and even cancel standing data sent or received without having to manage multiple connections. Hence, to get the most of a QUIC connection this feature is important. When it comes to SBI, there are definitely cases where one NF consumer will have number of multiple standing requests to one of the NF providers. The HTTP/3 and QUIC framing and multiplexing provides essential support to perform the task efficiently. The efficiency gain in HTTP/3 and QUIC, compared to HTTP/2 over TLS/TCP, exists only when the transport connection is subject to packet loss. This is when TCP's head of line blocking will not allow releasing received data to higher layer, even if the data is completely received for independent HTTP/2 requests or responses.

### 5.4.3 Encrypted and Integrity Protected Transport details

Encryption and integrity protection are a very important aspect of the whole SBI concept. 3GPP has mandated the support of TLS 1.2 or 1.3 for 5G core. The NF provider needs to be sure about the identity of the NF consumer before serving. Even though a higher layer authentication could serve the purpose of the NF consumer authentication, protection in the transport layer can be very important for a PLMN to function properly. Moreover, the SBI opens up the possibility to run packet core in a general-purpose cloud environment where the communication between inter PLMN and intra PLMN need to be secure and integrity protected.

Having transport layer integrity and authentication mechanism the transport protocol is harden against both attacks and random corruptions that could affect the transport protocols action. For example, cryptographic integrity protection is many magnitudes better at detecting modifications of the transport protocol packet. Cryptographic integrity protection captures packet modifications of both random types, as well as hostile modifications, where checksums only have a weaker protection against random errors. This improvement prevents fouling up the transport protocol state, affecting performance and loss recovery. Data errors in IP/TCP packets that the TCP checksum fails to detect will result in the corrupt data being passed to TLS. TLS integrity verification of the TLS record will detect this corruption. Normally in this case, there are no alternative to terminating the TCP connection and restart it and re-issue the outstanding HTTP requests. QUIC's integrity verification will in this case only result in the individual packet being dropped, request retransmission, and not affect the connection state.

### 5.4.4 Connection setup improvements

QUIC can achieve faster connection establishment compared to TCP+TLS combination. The applicability of this features depends on the model is used for inter-NF communication. When long lived connection is used in between NFs which uses SBI for communication even if QUIC provides faster connection it will not impact the performance of the inter-NF communication significantly as only the initial request for a connection will see any improvement. However, if short lived connection models are used where NF-NF connection will be created for each request and response pair, QUIC will provide a faster experience of executing task via HTTP request/response as one or two RTT are saved. Overall the faster connection setup time will provide faster connection between NFs in start-up and/or recovery phase while lots of connection supposed to be made at once or in parallel.

### 5.4.5 Connection ID and Connection Migration

The connection ID provides certain flexibility in how the implementers realize front-end load-balancers for QUIC as the QUIC connection is not bound to 5 tuples (protocols and ports). In the case of SBI, both for cloud native implementation or bare metal implementation, this connection ID will provide the ability to establish network interface agonistic connection and move the connect between the interfaces as required without terminating the QUIC connection.

Server-side migration is currently only specified to be done shortly after connection handshake using the Server Preferred Address mechanism discussed in Clause 9.6 of IETF RFC 9000 [5]. This mechanism requests that the client sends the packet destined to the server to this preferred address instead of the original one. Future versions or extensions may specify mid connection server side migration.

Client-side migration may occur at any point after the handshake has completed. This can be done intentionally by the client when another network interface has become available, where it first probes the new path from this other interface to the server, and after path verification starts using non-probing packet, thus completing the migration. It can also occur implicit, due to a NAT rebinding where the server-side observable source address and port has changed due to this rebinding. Here the use of non-probing packets results in immediate path migration to the new path, and at the same time the server initiates a path validation.

### 5.4.6 Improved Recovery and Acknowledgement

When the transport connection is subject to packet loss, QUIC recovery and acknowledgment mechanisms allow to optimally retransmit the lost frames and to avoid unnecessary retransmissions of data that have reached the receiver.

## 5.5 Features of QUIC Not Applicable to 3GPP SBI

### 5.5.1 General

This clause reviews the features of QUIC that potentially are not applicable to 3GPP SBI and why they are not applicable.

### 5.5.2 0-RTT DATA

0-RTT Data has very limited applicability to 3GPP SBI for several reasons. The foremost is the security properties of 0-RTT data. As the 0-RTT data is protected using a Pre-Shared Key (PSK) and not a connection specific established state, the 0-RTT data is possible to replay by an adversary. It is also does not have full forward secrecy, i.e. if the PSK key is later compromised, then this message can be decrypted at that point.

The possibility for replay has multiple impacts. If the HTTP request in the 0-RTT data was not idempotent then the state of the NF could be changed (so IETF RFC 9001 [6] specifies that "0-RTT MUST NOT carry a self-contained trigger for any non-idempotent action"). Secondly, when replaying the order of requests can be changed by an adversary. Thus, changing the effect of them, e.g. moving a delete after a create. If multiple replays are allowed additional attacks are possible, including timing and measurement to attempt to determine other state. Overload concerns are also present both on the server side, as well as using 0-RTT as a method for amplifying the amount of data a spoofed source address attack results in.

Due to that many 3GPP SBI requests are not idempotent the potential use of 0-RTT data is very limited. By not allowing its use at all several vulnerabilities are avoided, resulting in a safer and less complex systems as no mitigations are needed.

In any case, given that the considerations on applicability of this feature are essentially related to security, it should be up to SA3 to determine whether it is recommended or not to use this feature in 3GPP networks; also, the recommendation may be dependent on intra vs inter -PLMN scenarios.

## 5.6 Comparison of Applicable Features with Rel-15 Transport

Table 5.6-1 provides a comparison of the features supported by HTTP/3 that are applicable to the 5GC SBI with HTTP/2 over TLS/TCP, based on the requirements from Transport Protocol for 3GPP 5GC SBI defined in clause 5.2 and additional evaluation criteria.

Table 5.6-1: Comparison of HTTP/3 and HTTP/2

|  |  |  |
| --- | --- | --- |
| Requirement/ Evaluation Criterion | HTTP/2 over TLS/TCP | HTTP/3 |
| R1. Reliable message delivery | TCP supports reliable and order-of-transmission delivery of data. | QUIC supports reliable and order-of-transmission delivery of data per stream. |
| R2. Flow control and congestion control mechanism | Flow control is supported at connection and stream levels.  TCP provides end-to-end congestion control, but with significant throughput reduction in case of packet loss. | Flow control is supported at connection and stream levels.  QUIC supports advanced mechanisms for loss detection recommending TCP NewReno for congestion control. Implementations may use other algorithms.  QUIC treats a Congestion Explicit Congestion Notification (ECN) in the IP header as a signal of congestion. |
| R3. Support of connection semantics | One HTTP connection maps to one TCP connection. | One HTTP connection maps to one QUIC connection. |
| R4. Failure to deliver one message shall not block subsequent messages | Head-Of-Line (HOL) blocking occurs if TCP segments get lost, delaying the delivery of all subsequent HTTP requests/responses until the lost segments are retransmitted. | QUIC avoids blocking the delivery of data for any other streams when a packet loss contains only part of a stream.  How much benefits this brings depends on how much packet loss occurs in the network. |
| R5. Transport protocol supports mechanisms to authenticate peer endpoint and to secure transfer of application messages | Authentication and secure transfer of application messages are provided by TLS (unless security is provided by other means).  NF service access authorization relies on OAuth2 using TLS. | QUIC uses TLS 1.3 for key establishment, but it has its own encryption and integrity layer that protects the QUIC packets.  NF access authorisation is FFS. |
| A1. Framing and Multiplexing | HTTP/2 supports multiplexing multiple parallel requests in separate streams in a non-blocking fashion (at HTTP level) over the same TCP connection.  See also R4 for HOL at TCP level. | HTTP/3 supports multiplexing of multiple parallel requests in separate streams in a non-blocking fashion over the same QUIC connection. |
| A2. Connection Setup Improvements | 1 ½ RTT is required to setup the TCP connection. 1 ½ RTT is required to setup the TLS connection.  Clients can start sending HTTP requests after 1 RTT (without using TLS) or 2 RTT (with TLS). | By combining connection setup and TLS handshakes, QUIC improves connection setup latency and security allowing 0-RTT connection setup. However, the precondition of the improvement is that the NF service consumer has had an earlier connection with the NF service provider so that it can reuse the earlier learnt connection settings including the security keys for 0-RTT. When using stateless services, no earlier connection to the same service instance can be assumed. In addition, if the connection is persistent the impact of 0-RTT connection setup is minimal to the overall performance. |
| A3. Failover to Alternate Path | TCP does not support multi-homing.  Failover to alternate paths can be supported by setting up additional TCP connections.  This requires the client to detect path failures (e.g. based on PING frames) and to switch to alternative TCP connection(s).  The server cannot send responses on an alternate path. | QUIC does not support multi-homing (*like supported by SCTP*).  QUIC supports client-side migration after the handshake has completed, and server-side migration shortly after the connection handshake, giving some flexibility to move the connection between interfaces without terminating the QUIC connection.  Failover to alternate paths can also be supported by using additional QUIC connections.  This requires the client to detect path failures (e.g. based on PING frames) and to migrate the connection to another network interface / local address, or to alternative QUIC connection(s).  There is no mechanism defined by QUIC that correlates heart beats and switching to failover paths; this has to be provided at the application layer.  The server cannot migrate the connection to a different network interface / local address during the lifetime of the connection.  The server cannot send responses on an alternative path, unless the client has migrated the connection on the client side. |
| A4. Low Response Time | Significant throughput reduction by TCP in overload and TCP head-of-line blocking are potential issues. See R4. | Throughput reduction due to congestion response for QUIC is similar to TCP's.  QUIC based transport avoids head of-line blocking. See R4. |
| A5. Scalability | FFS | FFS |
| A6. Time of Availability of used standards | Already available. | Planned completion by July 2019. |
| A7. Ease of troubleshooting and Monitoring | Many tools exist to trace/monitor HTTP REST APIs.  If TLS end-to-end encryption is used, this renders centralized logging at intermediates impossible.  An HTTP response follows the same path as its request as it is sent on the same TCP connection. | Many tools exist to trace/monitor HTTP REST APIs, but less widespread support for QUIC so far.  QUIC requires end-to-end encryption that would render centralized logging at intermediates impossible or much more complex. QUIC exposes a spin bit in short packet header to measure the RTT of a connection.  An HTTP response follows the same path as its request as it is sent on the same QUIC connection. |
| A8. Ease of traversal of carrier-grade ALG/NAT/firewall | Need to configure operator-grade firewalls to pass TCP/TLS/HTTP. For bidirectional communication, configuration for two connections may be required, but security gateways can reduce the number of required connections (see 3GPP TS 33.210 [24]). | Need to configure operator-grade firewalls to pass UDP/QUIC. For bidirectional communication, configuration for two connections may be required, but security gateways can reduce the number of required connections (see 3GPP TS 33.210 [24]). |
| A9. Impacts to GSMA GRX/IPX | Support being defined for Rel-15  (see GSMA NG 113 [25]). | No HTTP/3 support so far. |
| A10. Use of proxies | HTTP/2 supports the use of proxies in the path. | HTTP/3 Proxy functionality still at very early stage (see clause 6.2). |
| A11. Idle HTTP connections | PING frames are used to test whether a connection is still alive. | PING frames are used to test whether a connection is still alive and to keep the connection alive.  QUIC endpoint declares an idle timeout during the handshake. If the connection remains idle (no packets received) for longer than the advertised idle timeout, the peer will assume that the connection has been closed. |
| A12. Availability of standard APIs (e.g. socket APIs) | Many libraries to choose from for HTTP/2. | QUIC support is not yet so widespread. |
| A.13 Stream IDs | HTTP/2 stream identifiers are coded with 31 bits. Stream IDs can exhaust during the lifetime of the HTTP/2 connection, which complexifies the management of connections. | QUIC stream identifiers are coded with 62 bits. Stream IDs exhaustion becomes nearly impossible during the lifetime of a QUIC connection, which simplifies the management of connections. |
| A14. Message Prioritization | Client and Server can set the priority of an HTTP request and response.  Multiple HTTP/2 connections between two HTTP/2 end points are necessary: one per DSCP value (see clause 6.8.8 of 3GPP TS 29.500 [4]), which complexifies the management of connections. | QUIC relies on receiving priority information from the application. QUIC frames do not allow to exchange priority information between peers.  Only the client is allowed to send HTTP/3 PRIORITY frames over the request or control stream.  There is an assumed integration between the HTTP/3 layer and QUIC to enable QUIC packet sender to prioritize the different streams based on what is configured on the HTTP/3 layer.  Multiple QUIC connections may still be necessary for messages with different SBI message priorities / DSCP to handle congestion control correctly, since using multiple DSCP on the same QUIC connection could confuse the congestion control algorithm, as it does not separate information on the different priority levels. The general problem is discussed in clause 5.1 of IETF RFC 7657 [29]. |

# 6 HTTP/3

## 6.1 Introduction

This clause will contain description about the mapping and usage of HTTP/3 including some of the not so well understood/documented aspects.

Figure 6.1-1 provides an overview of the HTTP/2 and HTTP/3 protocol stacks, highlighting key features of the HTTP and transport layers. See clause 5.3 for a detailed description of the QUIC features.



Figure 6.1-1 HTTP/3 vs. HTTP/2 protocol stacks

With HTTP/2 protocol stack, TLS is used conditionally when the security is not provided by other means.

With HTTP/3 protocol stack, TLS is integrated into QUIC and is used always, i.e. TLS is mandatory.

## 6.2 HTTP/3 Proxies

### 6.2.1 General

HTTP clients can be configured to route their outgoing HTTP requests via a HTTP proxy. If the NF service consumer (i.e. HTTP client) is configured to route its message via a HTTP proxy, the NF service consumer will try to setup a transport connection towards the proxy. If the NF service consumer knows that the proxy supports QUIC based on configuration or other offline means, the transport connection towards the HTTP proxy may use QUIC. Thereafter how the HTTP/3 proxy further communicates with the NF service producer for various scenarios are explained in the clauses below.



Figure 6.2.1-1 NF Service Consumer to NF Service Producer Communication with HTTP/3 Proxy on Path

### 6.2.2 When NF Service Consumer Side Uses QUIC

#### 6.2.2.1 Case A: Invoking http API Supporting Only TCP Transport

This case is not describe in IETF RFC 9114 [7].

In this scenario:

- NF service consumer supports QUIC and has established a QUIC transport connection with its next hop HTTP proxy;

- NF service consumer discovers that NF service producer supports only TCP.

- The URI scheme of the API exposed by the NF service producer is http

In this case, the NF service consumer has the following options:

- Option#1: The NF service consumer uses TCP transport towards the proxy as well. This implies the proxy also supports TCP transport (which is a reasonable assumption considering that during the migration from TCP to QUIC many HTTP entities will support both transports).

- Option#2: The NF service consumer uses QUIC transport towards the HTTP proxy and the proxy uses TCP transport towards the NF service producer. The HTTP proxy discovers whether the NF service producer supports TCP or QUIC based on apriori connection setup. For example, in the case of SEPP all NFs in a PLMN connect to the SEPP and establish a HTTP/2 or HTTP/3 connection depending on what transport is supported by both the SEPP and the NF service producer. IETF RFC 9114 [7], clause 3.2 specifies that HTTP/3 clients shall indicate the target domain name during the TLS handshake of QUIC connection setup. The certificate provided at connection setup shall be valid for the target domain name.

Editor's Notes: It is unclear what domain name shall be used for the target domain name when the connection is with a proxy (proxy domain name or the origin server one).

The draft also says in clause 3.3 that a connection to a server endpoint may be reused for requests with multiple different URI authority components. The client may send any requests for which the client considers the server (the one at the existing connection endpoint) authoritative.

Editor's Notes: In our case the client knows that existing QUIC connection ends at a proxy and not at a server. So it is unclear if we can reuse an existing QUIC connection to a proxy endpoint. Also it is unclear if a client can consider a proxy as an authoritative server as proxies and servers are essentially different HTTP entities.

The RFC specifies in clause 3.3 how the client knows that the server at the endpoint of the reused QUIC connection (the proxy in our case) is authoritative for requests directed to other domains. It mentions that typically the client discovers that a particular server is the authoritative HTTP/3 endpoint based on the client having received Alt-Svc HTTP response header or the HTTP/2 ALTSVC frame (see IETF RFC 7838 [20]).

Editor's Note: Whether other mechanisms other than use of IETF RFC 7838 [20] can be considered to discover a particular HTTP/3 endpoint is the authoritative endpoint for a URI authoritative component is FFS.

Finally, the clients shall check that the nominated server can present a valid certificate for the Origin Server before considering it authoritative. Therefore, the HTTP proxy has to present a certificate to the HTTP/3 client on behalf of the HTTP Origin Server (NF service producer) that is valid for multiple domain names and signed by the client network's own certificate authority. In roaming, the client network owner (the VPLMN) and the origin server network owner (the HPLMN) are different authorities and such a certificate is impossible to issue by a regular certification authority (e.g. Verisign). The only possibility is that the HTTP client should be configured to trust the HTTP proxy as the certificate authority. Only then this option#2 will work.

- Option#3: The NF service consumer uses QUIC transport towards the HTTP proxy. The proxy provides a certificate only valid for itself at QUIC connection setup. When the NF service consumer needs to send a request to an NF Service producer it first establishes a tunnel through the proxy by sending an HTTP CONNECT message in a new stream with an ":authority" pseudo-header field identifying the NF Service producer. The proxy then creates a TCP connection towards the NF service producer. Once the TCP connection is completed, a tunnel is created between the NF service consumer and producer. This tunnel is used by the NF service consumer to create a direct HTTP/2 connection (without an end to end TLS) with the NF service producer. HTTP/2 messages can now flow between the two entities. This is illustrated by the figure below.



Figure 6.2.2.1-1: http via HTTP/3 Proxy to NF Service Producer Supporting TCP

NOTE 1: Option 3 is not described by IETF RFC 9114 [7] which only describes the use of the CONNECT method to setup a TLS session between an HTTP client and an Origin server.Most of the existing implementation also restricts the usage of CONNECT to https URIs. This option excludes the use of current implementations available on the market. However for 3GPP NF services, the HTTP clients will be the HTTP client libraries supported in various programming languages. One could program in such a way to use HTTP CONNECT via a proxy for http URI too.

NOTE 2: IETF RFC 9114 [7] doesn't explicitly say if the verifications listed in clause 3.3 of the RFC that authorize the reuse of an existing QUIC connection are applicable to the CONNECT method.

#### 6.2.2.2 Case B: Invoking http API Supporting QUIC Transport

In this scenario:

- NF service consumer supports QUIC and has established a QUIC transport connection with its next hop HTTP proxy;

- NF service consumer discovers that NF service producer also supports QUIC.

- The URI scheme of the API exposed by the NF service producer is http

In this case the NF service consumer uses QUIC transport towards the HTTP proxy and the HTTP proxy also uses QUIC transport towards the NF service producer. The QUIC connections on the both side of the leg of HTTP proxy are separate QUIC connections. As TLS is integrated in QUIC, this means this setup would also terminate the TLS at the proxy which is undesirable. In case of proxying with HTTP/2 only the TCP connection is terminated at the proxy but the TLS connection on top of TCP is end-to-end.

The figure below illustrates the case where the HTTP client and server are connected with two QUIC connections through an HTTP proxy.

The connection with the HTTP proxy would be reused for requests sent to multiple domains. When the proxy needs to forward a message to a new HTTP server, it establishes a new QUIC connection with it. The server provides a valid certificate for itself.



Figure 6.2.2.2-1: http via HTTP/3 Proxy to NF Service Producer Supporting QUIC

Case B is not described in IETF RFC 9114 [7] and the same questions regarding the QUIC connection with the proxy as specified for Case A remains open with Case B.

As per IETF RFC 9114 [7], clause 3.3, a HTTP client MUST verify if the nominated HTTP server it is communicating with (i.e. HTTP proxy in this case) can present a valid certificate for the origin before considering it authoritative. Hence in order to setup an end to end QUIC connection between the HTTP client and the HTTP server via a HTTP/3 proxy, an equivalent of HTTP CONNECT to setup a tunnel is required. Currently such an option does not exist. HTTP CONNECT is used only when the URI scheme is https and upon getting a HTTP CONNECT request a HTTP/3 proxy establishes a TCP connection with the HTTP server (and not a QUIC connection) as specified in clause 4.4 of IETF RFC 9114 [7].

NOTE: The use of HTTP CONNECT by HTTP clients when accessing https URI via a proxy is not mandated in IETF RFC 7231 [22]. However many browsers by default use HTTP CONNECT when accessing https URIs via a proxy. For 3GPP NF services, the HTTP clients will be the HTTP client libraries supported in various programming languages. One could program in such a way not to use HTTP CONNECT via a proxy and trust the certificates issued by the proxy effectively allowing the proxy to act as man in the middle.

IETF draft-pardue-httpbis-http-network-tunnelling-01 [21] tries to provide a solution that permits a UDP-based HTTP/3 client behind an HTTP proxy to establish an HTTP/3 session with the origin. As the successor approach, IETF is reviewing a working group formation proposal to work on a HTTP based proxying solution for end to end encrypted traffic, named MASQUE, see IETF draft-ietf-masque-connect-ip-13 [35]. This result of this work will allow the end point to communicate with end to end QUIC encryption while use the proxy on the path. This means the HTTP client (consumer) will maintain a QUIC tunnelling connection towards the HTTP/3 proxy and inside that tunnel the consumer will have an end to end QUIC connection towards the HTTP server (provider).

#### 6.2.2.3 Case C: Invoking https API Supporting Only TCP Transport

In this scenario:

- NF service consumer supports QUIC and has established a QUIC transport connection with its next hop HTTP proxy;

- NF service consumer discovers that NF service producer also supports only TCP.

- The URI scheme of the API exposed by the NF service producer is https

In this case the following sequence of events happen

- HTTP client establishes a QUIC connection with the HTTP proxy, if not setup earlier.

- HTTP client sends a HTTP CONNECT request to the proxy with ":authority" pseudo-header set to the NF service producer FQDN or IP address.

- HTTP proxy sets up a TCP connection with NF service producer (HTTP server).

- HTTP proxy sends a HTTP CONNECT response to the HTTP client.

- HTTP client does end to end TLS connection setup with the NF service producer. An encrypted tunnel between the client and the server is now setup and HTTP/2 connection can be setup on top.

NOTE 1: The HTTP client has to do encryption twice - one for the TLS tunnel and one for the QUIC connection with proxy.

NOTE 2: The current design of CONNECT-based tunnelling reserves an ordered byte stream (HTTP/2 and HTTP/3) for the client-to-proxy hop. This is subject to head of-line (HoL) blocking. See IETF draft-pardue-httpbis-http-network-tunnelling-01 [21] clause 3.6.

This scenario is illustrated in the figure below



Figure 6.2.2.3-1: https via HTTP/3 Proxy to NF Service Producer Supporting TCP

According to RFC 7230 [23] clause 2.7.3, the client shall ensure that its connection to the origin server is secured through the use of strong encryption, end-to-end, prior to sending the first HTTP request when the https URI scheme is used.

When an HTTP proxy is deployed, end-to-end security is ensured by setting-up a tunnel between the client and the Origin server using the HTTP CONNECT method which is then secured with TLS.

A HTTP client implementation may decide not to enforce E2E security with TLS though the https URI scheme is used and connection to the Origin server is done via a proxy. IETF RFC 7231 [22] does not mandate the use of HTTP CONNECT for accessing https URI via a proxy. If a HTTP client decides not to use CONNECT, then it may trust the certificates issued by the HTTP/3 proxy on behalf of the HTTP/TCP server signed by the proxy's certificate authority, thus allowing the HTTP/3 proxy to act as man in the middle. This would violate the requirement for the HTTP client in RFC 7230 [23] clause 2.7.3.

Alternatively the NF service consumer may decide to use TCP transport towards the HTTP/proxy similar to option#1 provided in clause 6.2.2.1. In this case, the NF service consumer avoids double ciphering.

#### 6.2.2.4 Case D: Invoking https API Supporting QUIC Transport

In this scenario:

- NF service consumer supports QUIC and has established a QUIC transport connection with its next hop HTTP proxy;

- NF service consumer discovers that NF service producer also supports QUIC.

- The URI scheme of the API exposed by the NF service producer is https

In this case the following sequence of events happen

- HTTP client establishes a QUIC connection with the HTTP proxy

- HTTP client sends a HTTP CONNECT to the proxy with URI set to the NF service producer API URI.

- As specified in IETF RFC 9114 [7] clause 4.4, the proxy establishes a TCP connection to the HTTP server. However it is desirable that a mechanism is available for the HTTP/3 proxy to instruct the use of QUIC connection to a HTTP server instead of TCP.

- Currently there is no mechanism that exists in IETF RFC 9114 [7] where a HTTP/3 proxy is instructed to use a QUIC connection to a HTTP server instead of TCP.

IETF draft-pardue-httpbis-http-network-tunnelling-01 [21] tries to provide a solution that permits a UDP-based HTTP/3 client behind an HTTP proxy to establish an HTTP/3 session with the origin. But at this moment this is an individual draft and is in very early stage.

According to RFC 7230 [23] clause 2.7.3, the client shall ensure that its connection to the origin server is secured through the use of strong encryption, end-to-end, prior to sending the first HTTP request when the https URI scheme is used.

A HTTP client implementation may decide not to enforce E2E security though the https URI scheme is used and connection to the Origin server is done via a proxy. IETF RFC 7231 [22] does not mandate the use of HTTP CONNECT for accessing https URI via a proxy. If a HTTP client decides not to use CONNECT, then it may trust the certificates issued by the HTTP/3 proxy on behalf of the HTTP/3 server signed by the proxy's certificate authority, thus allowing the HTTP/3 proxy to act as man in the middle. This would violate the requirement for the HTTP client in RFC 7230 [23] clause 2.7.3.

### 6.2.3 When NF Service Consumer Side Uses TCP

#### 6.2.3.1 Invoking http API Supporting QUIC Transport

In this scenario:

- NF service consumer supports only TCP and has established a TCP transport connection with its next hop HTTP proxy;

- NF service consumer discovers that NF service producer supports QUIC.

- The URI scheme of the API exposed by the NF service producer is http

In this case the HTTP proxy has to act as a HTTP/TCP proxy on one side and as a HTTP/3 client on the other side. The proxy simply relays the message received on TCP connection to the QUIC connection. But in order for the proxy to setup a QUIC connection with the NF service producer, the proxy has to discover that the NF service producer supports QUIC. This can be achieved by using solution described in clause 8.2.2. This means the NF service producer also should support TCP.

#### 6.2.3.2 Invoking https API Supporting QUIC Transport

In this scenario:

- NF service consumer supports only TCP and has established a TCP transport connection with its next hop HTTP proxy;

- NF service consumer discovers that NF service producer supports QUIC.

- The URI scheme of the API exposed by the NF service producer is https

When https scheme is used, the HTTP client first sends a HTTP CONNECT request to the HTTP proxy. However as per IETF RFC 9114 [7], clause 4.4 and IETF RFC 7231 [22], clause 4.3.6, when a HTTP proxy receives a HTTP CONNECT method, it establishes a TCP based tunnel towards the NF service producer (the HTTP destination origin server) so that a TLS connection end to end from the HTTP client to the HTTP destination origin server can be setup. Since the semantics of HTTP CONNECT demands this, the HTTP proxy will not use QUIC towards the HTTP server (NF service producer) even if it supports QUIC.

A HTTP client implementation may decide not to use HTTP CONNECT to access a https URI via a proxy. IETF RFC 7231 [22] does not mandate the use of HTTP CONNECT for accessing https URI via a proxy. If a HTTP client decides not to use CONNECT, then it may trust the certificates issued by the HTTP/3 proxy on behalf of the HTTP/TCP server signed by the proxy's certificate authority, thus allowing the HTTP/3 proxy to act as man in the middle. This would violate the requirement for the HTTP client in RFC 7230 [23] clause 2.7.3.

## 6.3 Considerations for HTTP/3

### 6.3.1 General

3GPP TS 29.500 [4] mandates HTTP/2 over TCP as protocols to be use for SBI. Running HTTP/3 requires special consideration as many of the HTTP/2 features can be taken care of by QUIC. HTTP/2 and QUIC contains similar features like stream, framing, multiplexing. Moving from HTTP/2 over TCP to HTTP/3 will require the application layer protocol behavior and implementation to be changed. Hence, it is important to identify the changes required both in HTTP/2 and QUIC implementations. This clause details the features and properties need special attention when HTTP/32 is used.

### 6.3.2 Connection setup and management

To use HTTP over QUIC requires explicit discovery of QUIC protocol support in the client and server. The server can advertise the support for the QUIC as a transport protocol then client can use some explicit information provided by the server or prior knowledge of the previous contact to the server to select QUIC as a transport protocol. Different alternatives to do the discovery of QUIC support in the NFs are discussed in clause 7.2.

QUIC connection level settings are communicated between client and server at the crypto handshake. However, the HTTP/3 specific settings (see IETF RFC 9114 [7]) are set via SETTINGs frame sent by the client and server via the HTTP/3 control stream after QUIC connection is established.

As QUIC allows stream multiplexing the HTTP clients can multiplex multiple HTTP/3 requests on to same QUIC connection as long as the server has the authority to serve the request. This reduces the need for multiple connections and improves performance by avoiding the time it takes to establish new connections. In case of SBI, every consuming NF will originate request to a specific provider NF. Hence, there will be one to one mapping between the server and origin. However, it is also possible to install a frontend proxy to hide a number of provider NFs that is managed by one administration. In this case the NF consumer will establish single connection towards the frontend proxy and multiplex request towards different NF providers over a single QUIC connection, treating the frontend proxy as a server endpoint.

### 6.3.3 Streams, framing and multiplexing

The QUIC stream number space is larger than that of HTTP/2. HTTP/3 uses a completely new framing concept. The HTTP/2 framing is completely changed, including the basic HTTP/2 Frame Header layout. Consequently HTTP/3 uses only the QUIC streams and does not have a stream Identifier in the HTTP/3 frame header..

Another important difference is the HTTP/3 only guarantee ordered delivery on the stream level while HTTP2 expects absolute ordering on the frames across multiple streams. HTTP/3 will break any such ordering assumption.

When HTTP runs over QUIC the HTTP layer does not require to do any stream multiplexing. QUIC maps each of its streams to a HTTP transaction. A stream is closed when the RESET\_STREAM is received in QUIC. In HTTP/2, a stream is half-closed when the frame with END\_STREAM bit set is received, and is closed when the RESET\_STREAM is received. The additional difference is that in the current specification, HTTP/3 does not use server initiated bidirectional stream. This means unidirectional streams are created from both the client and server with indication of the purpose of the stream as stream header at the beginning of the streams. All client initiated bi-directional streams are used for requests and responses.

Editor's Note: Some aspects of connection management are still not clearly defined in the IETF drafts, such as the usage of client-initiated vs. server-initiated streams, and bi-directional vs. unidirectional streams. A more accurate description and analysis of these aspects is FFS, once the IETF drafts are further developed.

The HTTP/3 frame type definition follows QUIC encoding concept. QUIC uses a variable length integer encoding which allows a larger number of stream IDs compared to HTTP/2 encoding. This change results in different HTTP/3 frame types and requires a mapping from HTTP/2 to HTTP/3 frame types. HTTP/3 (see IETF RFC 9114 [7]) defines this mapping of HTTP/2 over HTTP/3.

### 6.3.4 Prioritization

In case of HTTP/2 the clients can set stream priority as defined in IETF RFC 7540 [13], at the creation of the stream and update the priority using PRIORITY frame. However, in case of HTTP/3, only the client is allowed send PRIORITY frames over control stream. The priority clause on the HEADER frame is removed.

### 6.3.5 Server Push

HTTP/3 uses a different server push mechanism than what is defined for HTTP/2 in IETF RFC 7540 [13]. HTTP/3 uses two new frames to accomplish server push – a) a modified PUSH\_PROMISE frame and b) a MAX\_PUSH\_ID. The modified PUSH\_PROMISE frame does not refer to a stream as originally designed in IETF RFC 7540 [13], it uses as PUSH\_ID that uniquely identifies a server push. HTTP/3 defines the use of PUSH\_IDs in PUSH\_PROMISE frame, DUPLICATE\_PUSH frame, CANCEL\_PUSH frame and PRIORITY\_FRAME in addition to their use in Push Stream headers (a unidirectional server initiated stream) for now. The server can only push, and it can initiate pushing only one it receives a MAX\_PUSH\_ID frame from the corresponding client. Details of the modified server push mechanism is described in IETF RFC 9114 [7].

### 6.3.6 Compression (HPACK vs QPACK)

The Header Compression for HTTP/2, HPACK (see IETF RFC 7541 [14]), provides compression of HTTP header fields. Significantly reducing the headers, especially for sequential HTTP/2 request responses to the same server, where repeated and redundant information is efficiently compressed. The use of HPACK in HTTP/2 is one of the more significant performance improvements compared to HTTP 1.0 or 1.1. HPACK was defined based on one important assumption, namely the TCP in-order delivery of the different HTTP/2 frame types across all the streams. Thus, the encoder knows in which order the decoder will receive and process the various frames, and how the decoder state will be updated. HTTP/3 does not provide the same deterministic and guaranteed in order delivery mechanism between different HTTP requests. HTTP/3 can avoid this head of line blocking and provide improved performance by delivering to higher layers the HTTP messages in the order they are successfully delivered to the peer. However, if one would use HPACK without modifications, this could result in the decoder blocking or producing the wrong output. Therefore, header Compression for HTTP/3, QPACK (see IETF RFC 9204 [10]), is being defined.

QPACK is a redesigned version of HPACK that can support out-of-order delivery. It allows flexibility in the encoder to perform trade-offs between compression ratios and likelihood of head of line blocking due to out of order delivery. The changes in QPACK allows for much reduced head of line blocking at similar compression efficiency for a given packet loss rate. It also provides the implementation freedom to select how robust the transaction should be against packet loss. This at the cost of requiring HTTP/3 servers to implement the new QPACK mechanism, even if some reuse of the HPACK implementation is possible.

# 7 Key Requirements for Supporting QUIC

## 7.1 Introduction

This clause contains the key requirements to be solved in order for QUIC to be considered as a transport replacement for 5GC SBI. For each of the key requirements listed below, solutions are described in clause 8.

## 7.2 Discovery of QUIC support

As Release 15 of 3GPP TS 29.500 [4] defines SBI implementation with protocol combination of HTTP over TCP, the deployment of QUIC as a replacement transport protocol for TCP will require a discovery method for the NF acting as HTTP client for both interoperability and backward compatibility. The discovery of a support of QUIC for a particular SBI must be done at the beginning of connection establishment and if QUIC is supported by the NF acting as server then QUIC should be used for all future communication towards that NF.

## 7.3 Discovery of NRF's Support for QUIC

One of the potential solutions for discovering the support of QUIC by an NF is to use to the NF discovery service of the NRF (see clause 8.2.1). In order for an NF service consumer to use the NF discovery service if the NRF, the NF service consumer should know the transport protocol capability of the NRF itself. This key issue will address the following aspects:

- How to discover the transport protocol capability of a NRF?

- How to provide the transport protocol capability of a remote PLMN NRF to a local PLMN NRF for home routed roaming scenarios?

## 7.4 Migration to QUIC

Since 5GC deployments based on release 15 will happen initially, any attempt to introduce NF service instances that support QUIC as transport protocol need to be carefully thought about in order to avoid interoperability issues. This key issue will study the following aspects with respect to the migration to QUIC.

- Steps to follow when adding NF service instances that support QUIC as transport protocol, into the network.

- Deployment topologies where NF service instances that support QUIC as transport protocol can be introduced.

- When the NF service consumers can use the added NF services supporting QUIC?

- Steps to follow to decommission NF services that support TCP, when required.

## 7.5 Support of Indirect Communication

Indirect Communication refers to the communication between a NF Service Consumer and an NF Service Producer using a Service Communication Proxy (SCP), in the Rel-16 Service Based Architecture.

Indirect communication may be supported (see 3GPP TS 23.501 [2]):

- without delegated discovery (Option C): Based on query result, the consumer does the selection. The consumer sends the request to the SCP containing the address of the selected producer (e.g. an NF instance or NF Service instance set), the SCP may select an instance and it routes the request to the targeted producer instance.

- with delegated discovery (Option D). Consumers do not do any discovery or selection. The consumer adds any necessary discovery and selection parameters required to find a suitable producer to the request. The SCP uses the request address and the discovery and selection parameters in the request message to route the request to a suitable producer instance.In both options, the SCP needs to access and optionally modify the contents of the HTTP messages exchanged between the NF Service Consumer and NF Service Producer. This entails that the HTTP/3 connection cannot be established end to end between the NF Service Consumer and NF Service provider. Instead, standalone HTTP/3 connections need to be set up between the NF Service Consumer and the SCP, between the SCP and the NF Service Producer, and between SCPs if there is more than one SCP between the NF Service Consumer and NF Service Producer.

This key issue will analyse how to support HTTP/3 for Indirect Communication.

NOTE: See clause 6.2 for a general description of HTTP/3 in presence of proxies on the path.

# 8 Solutions for Key Requirements

## 8.1 Introduction

This clause contains the solutions for the key requirements identified in clause 7.

## 8.2 Solutions for Discovery of QUIC support

### 8.2.1 Using the Discovery Service of the NRF

Using the Discovery Service of the NF Repository Function (See 3GPP TS 29.510 [19]) is a possible solution for discovering if a NF instance's SBI supports using QUIC as transport protocol.

When a consumer is performing service discovery of NF instances for a service, it will also learn which if any of the instance support using QUIC by having IpEndPoint definitions in the NFProfile with the transport protocol set to QUIC.

This solution enables the NF consumer to know of the support even prior to attempting to establish a transport connection to the NF producer. The consumer has to use the NRF service discovery irrespectively of the determination of QUIC support.

The realization of this solution requires definition of the QUIC as TransportProtocol in the NFProfile, see clause 6.1.6.3.5 of 3GPP TS 29.510 [19].

The solution will have to determine if this discovery should be only for a general support of QUIC independent of version or if also all the versions should be encoded in the NFProfile.

A QUIC supporting NF can potentially support several different versions of QUIC, thus efficient enumeration of versions would be desired.

However, assuming that at least one mandatory to implement version of QUIC will be defined for SBA, it might be sufficient to simply indicate that QUIC in general is supported and rely on the version negotiation mechanism built in in QUIC as that would also avoid any interoperability issues.

For consumers of NF services that specifically want to determine which instances that support QUIC, extending the set of query parameters defined for the Nnrf\_NFDiscovery Service API (clause 6.2.3.2.3.1 of 3GPP TS 29.510 [19]) will be needed.

A parameter such as transport-protocol which takes an array of protocol enumerations would solve this issue.

Here enumerating individual QUIC versions would create some extra complexity.

### 8.2.2 Using Alt-Svc Header

The current QUIC working group draft on HTTP over QUIC (HTTP/3) (See IETF RFC 9114 [7]) defines a discovery method of QUIC support using Alt-Svc HTTP response header defined in IETF RFC 7838 [20]. In this case the NF as HTTP server can notify the NF as HTTP client about the support of QUIC protocol with a HTTP response header with any HTTP response. An example of such response will look like below:

HTTP/1.1 200 OK

Content-Type: text/html

Alt-Svc: h3=":50443";quic="1,1abadaba"

Here, the " h3" is the ALPN token identifies HTTP/3 and "quic" is a new parameter defined to advertise the versions supported by the NF. The syntax of Alt-Svc is defined in IETF RFC 7838 [20] and the "quic" parameter for Alt-Svc header to provide the QUIC version hints, is defined in HTTP/3 IETF draft (See IETF RFC 9114 [7]).

In this method, the HTTP client acting as NF consumer needs to start connection using TCP for the first contact with a HTTP server acting as NF provider. If the HTTP server response includes the Alt-Svc header then the HTTP client will re-establish HTTP connection over QUIC and save the protocol preference for further connection. After new QUIC connection established towards the HTTP server, the HTTP client must send all the requests over QUIC connection. The HTTP client then can terminate the previously established TCP connection.

As described, the downside of this method is that the HTTP client for the first contact with a HTTP server has to establish TCP connection to discover the QUIC support and terminate the already establish TCP connection. However, this should be only one-time event after discovering that one HTTP server supports QUIC the client must not repeat this discovery event.

This method allows a gradual deployment of QUIC in the PLMNs and does not require extra information exchange at the NF service discovery phase.

This solution requires that the HTTP server (NF Service Producer) can be reached over TCP in addition to QUIC, so a server supporting only QUIC would need additional mechanisms to let NF Service Consumers discover such support.

## 8.3 Solutions for Discovery of NRF's Support for QUIC

### 8.3.1 Providing NRF Transport Capability from NSSF

When network slicing is used, the NSSF is queried during

- Registration procedure; and/or

- PDU session establishment procedure

to select a network slice instance for the UE / PDU session. The NSSF may optionally return the network slice instance information (NsiInformation) which contains the NRF URI to be used for that network slice instance. In this solution it is proposed to return the NRF's transport capability also as part of the NsiInformation so that the NF service consumer that gets the NRF information from the NSSF knows which transport protocol(s) can be used towards the NRF.

The NSSF is provided the transport capability of the NRF when OAM configures the network slice instance information into the NSSF.

### 8.3.2 Providing Remote PLMN NRF's Transport Capability during NF Discovery

For home routed scenarios, the NRF in VPLMN contacts the NRF in HPLMN, via the SEPP. The issue of how HTTP requests over QUIC are routed when there is a proxy in between (i.e. if SEPP acts as a proxy) are specified in clause 6.2.2. Irrespective of the kind of solutions available for this issue, it may be required for the NF service consumer in VPLMN to provide the transport capability of the HNRF to the VNRF. This solution proposes the following:

- An NF service consumer in VPLMN discovers the NRF to use in VPLMN and the HPLMN for home routed scenarios by querying the NSSF or by local configuration. If the NSSF is queried, the NF service consumer obtains the transport capability of the VNRF and the HNRF from the NSSF as specified in subcaluse 8.3.1.

NOTE: As specified in clause 4.3.2.2.3.3 of 3GPP TS 23.502 [3], the discovery of VSMF/VNRF and HSMF/HNRF happen as separate procedures. The Nnssf\_NSSelection\_Get service is invoked twice in this case.

- Once the transport capability of both the VNRF and the HNRF are obtained, the NF service consumer knows which transport to use towards the VNRF.

- When the NF service consumer invokes the NF discovery service of the VNRF, it also provides as input the transport capability of the HNRF.

- The VNRF uses this information together with the information about the transport capabilities of the SEPP, taking into consideration the scenarios specified in clause 6.2.2, to decide which transport to use for contacting the HNRF.

### 8.3.3 Discovery Based On Local Configuration

In this solution, the transport capability of the NRF is discovered based on local configuration at the NF service consumer. This solution can be used when network slicing is not used or during scenarios where the NSSF is not involved (e.g. AMF does network slice selection based on local configuration).

## 8.4 Solutions for Migration to QUIC

### 8.4.1 Deployment Topologies to Introduce NF Services with QUIC Support

As identified in clause 6.2, HTTP/3 message traversal for http scheme APIs when a HTTP/3 proxy is involved on path is not yet clearly addressed in IETF. Similarly for https scheme APIs, the use of HTTP CONNECT from the HTTP client to the HTTP proxy, creates a TCP connection from the HTTP proxy to the NF service acting as HTTP server resulting in an end to end TLS connection from the HTTP client to the HTTP server. In this case also the presence of HTTP proxy on path implies that an NF service acting as server cannot use QUIC for https scheme APIs, until alternate mechanisms as discussed in IETF in IETF draft-pardue-httpbis-http-network-tunnelling-01 [21] reach some maturity.

Considering this the following are the deployment topologies where NF services with QUIC support can be introduced into a network without causing any issues in working towards a HTTP client.

- Intra PLMN NF service communication without any HTTP proxy as intermediaries.

- QUIC between HTTP client and HTTP proxy while TCP is used between HTTP proxy and the HTTP server (see Option#3, clause 6.2.2.1 and clause 6.2.2.3).

Editor's Note: The benefit of using QUIC with HTTP proxy and TCP on the other side is FFS.

For inter PLMN HTTP/3 messaging, SEPP is involved and the transport connection will have to terminate at SEPP. Hence an NF service consumer at VPLMN need to only consider the transport capabilities of SEPP and not the transport capabilities of the NF service producer in HPLMN. Irrespective of whether the NF service producer in HPLMN supports TCP or QUIC, as long as the NF service consumer in the VPLMN understands the API version of the NF service producer, it uses the transport protocol that it supports towards the SEPP in VPLMN.

### 8.4.2 Steps to Follow When Introducing NF Services with QUIC Support

The following steps have to be followed while introducing NF Services with QUIC support in a PLMN.

- The support for QUIC of the NF services shall be discoverable using one or any of the mechanisms mentioned in clause 8.2.

- Existing, NF service consumers that are already using TCP towards other NF service providers shall continue to use TCP, until they are upgraded to support QUIC.

- NF services that support QUIC, may also support TCP in order to interwork with existing TCP based NF service consumers.

### 8.4.3 Use of QUIC by NF Service Consumers

An NF service consumer that supports QUIC can use QUIC towards NF service producers that support QUIC after ensuring the following:

- The support of QUIC by the NF service producer is discoverable.

- There are no HTTP proxies on path or the HTTP proxy on path supports QUIC and HTTP CONNECT.

### 8.4.4 Decommissioning TCP

When all the NF services in a PLMN are upgraded to support QUIC a PLMN may consider to decommission the use of TCP transport. The following steps need to be followed while decommissioning

- Ensure that no NF service consumer is still using TCP towards the NF service for which TCP is to be decommissioned in that PLMN.

- If TCP is to be decommissioned for use by a SEPP in a PLMN, then ensure that all the other SEPPs it is interacting with support QUIC and there are no IPX on path.

- No NF service instance in the PLMN is using the solution described in clause 8.2.2 for the discovery of QUIC support.

It is recommended that TCP is not decommissioned until it is identified that there is no need for it within a PLMN.

# 9 Impacts to Service Based Architecture

## 9.1 Introduction

This clause identifies the impacts of introducing QUIC to service based architecture for which solutions can't be provided in 3GPP scope.

## 9.2 HTTP Proxy Traversal

As described in clause 6.2, IETF has not yet clearly specified how HTTP/3 works when there are proxies on path. The current drafts only define the use of HTTP CONNECT method from the client to the proxy and TCP thereafter. Hence at least until a solution for the use of hop by hop QUIC when HTTP proxies are involved is well specified in IETF, QUIC cannot be used in deployments where HTTP proxies are acting as intermediaries between NF services.

Editor's Note: Other impacts are FFS.

Editor's Note: Whether a specific action be sent to IETF is FFS.

## 9.3 QUIC's Security Mechanisms

QUIC as currently specified do not support any unencrypted mode, nor unauthenticated. This have several implications on the SBA architecture. Some of them have already been touched upon before.

- There is a requirement (see clause 3.2 of IETF RFC 9114 [7]) of explicitly addressing the QUIC peer when establishing a connection and include the target domain in the TLS handshake using SNI or other mechanism. This prevents any type of transparent HTTP proxies, and the next hop must always be known by the client.

- Use of any HTTP proxy will require additional functionality as discussed in Clause 9.2 and where only the one type of proxying, i.e. the use HTTP Connect method to establish end-to-end TLS connections over TCP from proxy to designated target domain. The other proxy cases discussed in Clause 6 lacks mature specifications. The current implication is that to enable QUIC in Release 16 SBA Architecture needs to be capable of operating without any HTTP proxies.

- Potential use of HTTP opportunistic security (IETF RFC 8164 [28]) for any "http" scheme requests over QUIC. This requires an additional QUIC connection as "http" scheme requests and "https" scheme requests are not allowed in the same connection (see clause 2.2 of IETF RFC 8164 [28]). It also requires additional HTTP server support to indicate this capability. This could enable QUIC migrations even if there are still some SBIs that doesn't support "https" scheme requests. However, if such interfaces exist it would be securer to upgrade them to support "https" scheme.

To summarize, with the current version of QUIC and HTTP/3 the 3GPP SBA architecture needs to assume that all requests and SBI will only use "https" scheme requests, that they will explicitly address the target NF, and that there is no possibility to deploy HTTP Proxies or other intermediaries acting above UDP layer in between NFs.

## 9.4 TCP Decommission in Migration Impacts Architecture

As discussed in Clause 8.4.4 there are certain requirements to enable decommissioning of TCP in a migration to QUIC. This also have certain architectural impacts. It requires a possibility to perform the first request to any NF using QUIC. The current specifications in IETF are not yet addressing such use cases, instead they assume that one will start with retrieving alt-svc information using HTTP over TCP and TLS. Thus, to support TCP decommission there need to be specified how one instructs a client to use QUIC directly to query without any prior alt-svc phase, primarily to issue requests to the NRF. The NRF can then likely function as solution for learning that QUIC is required for the other NFs.

## 9.5 Transport Proxy Traversal

TCP transport proxies may be deployed on the path between HTTP clients and servers in order to optimize transport on heterogeneous networks (e.g. satellite networks). They operate the flows based on TCP headers information without interfering with the HTTPS layer. QUIC breaks this layering as the transport and the HTTPS parts cannot be dissociated. As such the transport optimization proxy function will not be possible with QUIC.

## 9.6 Impacts on Troubleshooting

### 9.6.1 Introduction

This clause aims at highlighting the main impacts of QUIC introduction on maintenance and troubleshooting at network level and application level.

Indeed, in order to improve end-to-end protection, TLS 1.3 and QUIC enforce the usage of ephemeral cipher suites and Perfect Forward Secrecy (PFS). The latter prevents an attacker that managed somehow to steal the private key of a server from decrypting the content of exchanges she/he captured in the past. In practice, the existing OAM (Operation And Maintenance) solutions which are designed to work for RSA cipher suites would hence not be able to troubleshoot PFS traffic. This clause also details the impacts of using QUIC-based signalling protocol stack between two 5G NFs for the cases introduced in annex E of 3GPP TS 23.501 [2].

The following three clauses point out the main identified impacts of QUIC on network level and HTTP/3 level troubleshooting.

### 9.6.2 QUIC keying impact

QUIC decryption is per design trickier than TLS, including TLS 1.3, because each QUIC connection has at least 3 keys (0-RTT key, 1-RTT key, packet number encryption key) and also because 1-RTT key may change dynamically.

### 9.6.3 Network level troubleshooting

The following impacts are foreseen for network level troubleshooting:

- QUIC replaces both TLS and TCP. One of the main differences indeed is that QUIC encrypts the transport header in addition to the payload, which is highly relevant for the network level troubleshooting matters. The existing Network OAM (Operation And Maintenance) solutions which are designed to make use and act on TCP headers would hence not be able to troubleshoot QUIC traffic and even less be easily adaptable to perform this task.

- Moreover, QUIC includes an optional measurement bit, named spinbit, which allows in-path probes to measure both the round trip delay and the decomposition of the delay on both sides of symmetrical path. This is directly usable for options A and B for monitoring and troubleshooting the performance of the network between two NFs. Its usage for the options C and D though requires additionally the mapping of the observations on both legs of the SCP.

- As of version 19, QUIC specifications do not support packet loss measurements. Indeed, packet loss measurements by probes located on the path requires multiple probes and has more limited scope, which hence makes it much harder and resource demanding. Its deployability has still to be demonstrated.

- So as to enable an in depth analysis of the performance (e.g. flow control, etc.) between a consumer NF and a producer NF, the decryption of the entire QUIC message is often required in order to read the transport parameters of the QUIC packet header. Contrary to HTTP/2 over TLS, this has the side effect of decrypting and revealing application layer information to network probes.

NOTE: QUIC level logging techniques are also possible for these purposes but are currently not mature enough and are not yet considered for inclusion in QUIC IETF specifications.

- As already mentioned, options C and D flow control troubleshooting would also require streams identifiers mapping in order to get an end to end view.

### 9.6.4 Application level troubleshooting

The identified impacts on application level troubleshooting are depicted hereinafter for each one of the four communication options defined in annex E of 3GPP TS 23.501 [2].

- Option A - Direct communication without NRF interaction:

Analysing the exchanges between a consumer NF and a producer NF requires the decryption of QUIC traffic. Debugging currently relies on the analysis of the exchanges captured by probes. As QUIC imposes the usage of PFS, the decryption of PFS traffic by a probe requires the communication of the set of session keys to the probe by one of the peers. This would hence increase the complexity of the OAM (Operation And Maintenance) solutions.

NOTE: TLS 1.3 also mandates the use of PFS. However, the main difference with QUIC / HTTP/3 is that TLS usage is not mandatory in the current version of the specifications for HTTP/2.

- Option B - Direct communication with NRF interaction;

Same as option A.

- Option C - Indirect communication without delegated discovery;

In this option the signalling path is made of at least two legs and QUIC is hence on at least one side of the SCP.

- When QUIC is used only in one leg of the communication path, the other side may be HTTP/2 in the clear, HTTP/2 encrypted over TLS or in a hop by hop tunnel like IPsec. When carried over TLS, HTTP/2 can use either TLS 1.2 or TLS 1.3 versions.

As the signalling path is made of at least two legs, application level troubleshooting by external tools would require the support of the cypher suites in use and the credentials of the two sessions so as to be able to decrypt the messages exchanged and the mapping of QUIC and HTTP/2 stream identifiers.

- Case where QUIC is used in both legs:

There is only one version of HTTP/2. Versioning is a feature of QUIC though. Consequently, the application level troubleshooting tools must be able to decode, map and synchronize exchanges of different versions of QUIC.

- Option D - Indirect communication with delegated discovery.

Same as option C.

# 10 Current Implementation and Maturity Status

## 10.1 Introduction

Being a newly developed transport protocol the viability of considering QUIC in 5GC service based interfaces has to be gauged by the current implementation and maturity status. This clause describes the current stats of QUIC from the best of the knowledge point of the view.

## 10.2 Implementation maturity

TCP implementations are very mature. Most of the features has been widely used and issues has been resolved in the implementations and standards. Right now, this cannot be said for QUIC implementations. As the specification has been mainly finished as IETF RFCs, the QUIC implementations are still new. IETF QUIC working group has given lots of emphasis on interoperability testing on QUIC features. Number of opensource projects of implementing QUIC has passed the interoperability testing. However, those opensource projects usually put emphasis on particular features of interest from the implementer hence cannot be treated as production ready or assuming to have all the featured required. It is expected that QUIC implementations will have a higher degree of issues caused by mistakes and errors in implementation.

## 10.3 Hardware offload support

A server's capability to handle certain amount of load can be improved by having protocol off-load function in network interfaces (NIC). For TCP, this include checksum off-loading, segmentation offloading, and crypto offloading. These functions improve the performance of the server. TCP sending operation's CPU utilization can be reduced 50 times if TCP segmentation offloading (TSO) is used compared to standard TCP configuration. While TCP offload functions are existing and available in numerous NICs, there is significant uncertainty about what functions that are possible to implement for QUIC and according to which road-maps they will be implemented in server hardware.

When it comes to crypto acceleration via hardware offload there are both large commonalities and differences between TCP and QUIC. Both are based on TLS, but QUICs protection of its frames are using a QUIC specific format, but for modern implementations both are likely to use the same crypto primitives, i.e. most likely AES-GCM (see IETF RFC 5288 [31]) that is highly performant in a number of more capable CPUs due optimized functions. However, any more dedicated crypto off-load may be missing for QUIC.

There has also been significant discussion in the QUIC working group mailing list about the cost of doing the packet number encryption and decryption. An encrypted packet number requires to be first decrypted before the correct initialization vector for the rest of the packet can be produced. This has been criticized in the QUIC WG for making off-load processing more difficult and require multiple stages.

## 10.4 UDP Performance in Operating System

The current QUIC implementation uses UDP socket APIs to send and receive QUIC traffic. Compared to TCP implementation in different operating systems, UDP implementation is not that performant. It has been reported in IETF QUIC working group mailing list that for the UDP send cost the CPU consumption could be up to 50% while crypto operation is less than 10%. Google has also reported that with possible optimization, they were able to reduce QUIC's CPU cost from 3.5 to 2 times that required for TLS/TCP when serving YouTube traffic (see "Developing and deploying a TCP replacement for the Web" [32]).

Experiments show that a standard TCP configuration gains almost 5x speed-up over UDP in terms of cycle cost (see "Optimizing UDP for content delivery" [33]). Such gains for TCP come from the support of TCP segmentation offload (TSO) that allows the application to send larger data packet than the path Maximum Transmission Unit (MTU). TSO requires hardware support from the NICs. However, Generic Segmentation (GSO) can also perform segmentation to emulate TSO. TCP has support from both TSO and GSO. Recent experiments show promising results for UDP segmentation offloading with hardware support (see "UDP segmentation offload" [34]). UDP GSO is also available for Linux kernel and can be used via interfaces but need extensive configurations and understanding of link layer properties. The TSO operates on sender side and create a specific profile of traffic in the network. Because of that optimization can be achieved at the receiver side by consolidating multiple MTU packet side into fewer large one. Large receive offload (LRO) and generic receive offload (GRO) is used to achieve this optimization for TCP. The same is not available for UDP now. There are reports on work in progress on UDP GRO. To achieve high performance over UDP socket for SBI (some of the SBI message sizes may be large, potentially up to a maximum of 2 MB), different optimized configurations are needed and those optimizations need to be available in different operating system.

This UDP performance issues can eventually be overcame. There are already several techniques like UDP GSO, GRO (Generic receive offload), zero copy that can help achieving performant UDP operation and networking. As these techniques are at their early stage, the UDP performance issues need to take into consideration when selecting transport protocol for service based interfaces.

# 11 Evaluation and Conclusion

## 11.1 Evaluation

### 11.1.1 Requirements for Service Based Interfaces

QUIC fulfils the transport requirements (REQ#1 to REQ#5) identified in clause 5.2 for the 5GC Service Based Interfaces.

Using HTTP/3 instead of HTTP/2 does not change the semantics of the NF Services and does not change the API. No changes are expected either on the OpenAPI specification to support HTTP/3.

Further study would be needed to define how to support Indirect Communication in the 5GC with Service Communication Proxies (SCP), but preliminary it seems the CONNECT UDP method and MASQUE proxy functionality [36] allows tunnelling trough proxies.

### 11.1.2 Expected improvements

Clause 5.4 identifies features of QUIC (HTTP/3) that would be applicable to the 5GC Service Based Interfaces; the following potential (performance) improvements are anticipated in comparison to HTTP/2:

1) QUIC allows to overcome HOL blocking from which HTTP/2 is suffering if a TCP packet is lost or becomes corrupted;

2) loss detection mechanisms of QUIC are using more accurate means to indicate lost bytes and RTT measurements resulting in assumedly more efficient recovery mechanism;

3) faster connection establishment compared to TLS/TCP (1 RTT instead of 2), for short lived connections; however, when using persistent connections, this will not lead to a performance improvement;

4) the connection may be migrated to a different network interface or local address by the client during the lifetime of the connection or by the server during the connection establishment.

### 11.1.3 Issues

Following issues are identified:

1) The QUIC layer is end-to-end encrypted and use of proxies between end-to-end QUIC connections is not sufficiently covered. There is no support for instance to support an end-to-end QUIC connection through a proxy using HTTP CONNECT. But the CONNECT UDP method and MASQUE proxy functionality [36] allows an HTTP client to create a tunnel for UDP communications through an HTTP server that acts as a proxy and this is defined also for the HTTP/3 Requests.

2) The QUIC layer is end-to-end encrypted and thus allows less accurate network monitoring capabilities than TCP. The Spin bit allows a limited passive connection observation as described in IETF RFC 9000 [5].

3) The server cannot migrate the connection to a different network interface / local address during the lifetime of the connection; this requires clients to still rely on alternate QUIC connections for failover to an alternate path; also solutions for ensuring availability and stability of connections (e.g. failover to a secondary path like supported in SCTP multi-homing) are not provided at the QUIC protocol layer but have to be implemented at the application layer similar to HTTP/2. The Multipath QUIC will potentially enable some of these functions, as explained in IETF draft-ietf-quic-multipath-04 [37].

4) IETF RFC 9002 [8] provides recommendations on congestion control (e.g. TCP NewReno). According to that, implementations may use other congestion control algorithms than TCP NewReno and endpoints MAY use different algorithms from one another. Some measurements have shown that QUIC can overtake TCP in terms of how much bandwidth it can take leading to unfairness even if the congestion algorithm is the same (see e.g. "Taking a Long Look at QUIC" [30]). This leads to possible concerns on how fairly QUIC traffic mixes with TCP traffic, as this would be the case e.g. during migration scenarios (see clause 8.4). According to RFC 9002 [8] QUIC does allow faster iteration of changes in congestion control (compared to TCP), which can lead to unfareness in some cases. The issue becomes more complicated when mixing different congestion algorithms and deserves more experimentation. In case this becomes an issue in the network deployments, the related algorithms can be evolved and improved easily and faster (due to QUIC implementation runs on user space).

### 11.1.4 Other considerations

Following considerations also apply:

1) QUIC (HTTP/3) requires extensive changes to the HTTP protocol stack (compared to HTTP/2), by moving and combining several features from the HTTP/3 and TLS layers into the QUIC layer. The header compression mechanism is re-designed to support out-of-order delivery. The application is also impacted, e.g. socket APIs of the HTTP layer, discovery of QUIC support, co-existence of HTTP/2 and HTTP/3 capable NFs.

2) QUIC (HTTP/3) security aspects need to be assessed by 3GPP SA3, e.g. options exist for connection establishment and connection migration, end to end security with proxies on the path, inter-PLMN signalling and N32 interface, use of OAuth with QUIC, etc.

3) Even though the definition of HTTP/3 and QUIC have proceeded to RFCs in IETF, the inputs from the field deployment best practices, concerning system optimization are not mature enough for adoption of the protocol in the 5GC for 5G-Advanced;

4) It is open when industry grade stacks (for use in 5GC) will be available. Existing implementations of QUIC are still immature and do not have the full QUIC features available (e.g. congestion control).

5) Due to experimental nature of current implementations there are yet no reference API which would be required for application level interoperability. The current existing implementations allow only basic proof of concepts testing but not load or stress testing of different features.

6) There are no standard APIs (e.g. socket APIs) for QUIC connection setup. Though there are different open source implementations of QUIC, there is a lack of uniformity in terms of the interface they expose towards application for QUIC connection setup and hence as of now implementation portability is a concern. Currently no QUIC API definition is included in the IETF QUIC WG charter.

## 11.2 Conclusion

QUIC provides some new promising features especially in terms of performance increase for current Internet type of traffic (web-based client-server). It should be carefully checked whether those improvements can also be harvested in 5G signalling environment. Intensive testing, however, is only possible when HTTP/3 and QUIC specifications and implementations are mature enough (see clauses 10 and 11.1 for the current implementation and maturity status). Therefore, the finalization of the work on QUIC major RFCs at IETF in May 2021 can only be considered as a first step in the evaluation of the feasibility of HTTP/3 (QUIC) for 5G control plane.

How to support HTTP/3 (QUIC) for indirect communications via SCP (enhanced Service Based Architecture specified in 3GPP Release 16) would need to be further studied.

Supporting QUIC (HTTP/3) for 5G control plane would cause extensive efforts (standardization efforts in 3GPP to finalize the study also including SA3 involvement and do corresponding normative work, changes to 5GC NFs implementations, testing), cause disruption to existing deployments based on HTTP/2, complicate interoperability (multiple HTTP versions, risks on implementation maturity). Besides, there is no evidence yet that HTTP/3 would bring significant performance improvements to 5GC signaling justifying these efforts. The low number of companies' contributions to the current 3GPP TR over the past years is also an indication of a limited interest in moving this work forward for the 5GC.

It is therefore recommended to not consider HTTP/3 (QUIC) as a basis for 5GC control plane signalling.

Annex A (informative):  
HTTP/2 vs. HTTP/3 testing findings and considerations

QUIC can achieve the same computational efficiency as TCP+TLS1.3 once various protocol mechanisms and system optimizations are in use [38]. The referred study was comparing only the transports used for HTTP versions, but the application (HTTP) part was excluded. In principle, the used transport protocol should not limit either HTTP versions by setting the hard caps due to transport specific issues. Relevant mechanisms and optimizations used in the microbenchmark includes delayed ACK [39], GSO (Generic Segmentation Offload) and the used MTU. Without any additional mechanisms or optimizations, QUIC performed over 2x better than TCP+TLS1.3, but in the real field deployments the optimizations for QUIC are not yet in place and this typically turns this performance relation to opposite figures.

The authors of [40] concluded that most performance discrepancies are a result of the developer’s design or the operator’s configuration, which include topics, such as congestion control, flow control, handling of multiple streams, packet size/MTU, and client validation of the 0-RTT function.

For video streaming [41], QUIC transport resulted 8% less errors in video requests and this could be one sign of QUIC's more resilience against HoL blockings.

Based on the measurements in [42], 0-RTT support yields a bit over 10% better time to first byte for HTTP/3 compared to HTTP/2, which lacks the support. In the same tests, variable results where sometimes HTTP/3 did perform better and vice versa.

Performance measurement is highly use case dependent, i.e., traffic profiles/patterns are stemming from there, and for HTTP/2, this is shown clearly by this study [43].

For a single resource or a small file size transmitting, the QUIC implementations outperformed TCP ones (due to improved handshaking among other things), but once the amount of transmitted data increases, the performance of both transport protocols is getting similar. For HTTP, there are also other things impacting to the overall performance and that is how resource priorization is used as shown in [44]. Again, this emphasizes the use case dependency and most of the available performance tests are based on web browsing from public sources where typically small, medium or large amount of resources were downloaded as a single HTTP page request. One of the main KPIs measured, was the serving time of HTTP request and/or time to first byte.

In HTTP/2, the sources of the Head-of-Line (HOL) Blocking can stem from different layers like TCP and TLS. HTTP/3 (QUIC) supports stream-awareness in transport layer, which improves the performance when one stream cannot block another one. However the QUIC’s HOL blocking prevention only works if there are multiple resource streams active at the same time [45]. The improved tolerance against HOL Blocking and even the 0-RTT based connection re-establishment benefit the HTTP/3 (over the HTTP/2), but these benefits start to have the practical impact(s) with worsening networking conditions and the existence of such conditions is questionable in the closed and well-engineered 5G core signalling network.

Annex B:  
Change history

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Change history** | | | | | | | |
| **Date** | **Meeting** | **TDoc** | **CR** | **Rev** | **Cat** | **Subject/Comment** | **New version** |
| 2018-07 | CT4#85bis | C4-185419 |  |  |  | Implementation of C4-185047 (TR skeleton), C4-185420, C4-185231, C4-185421, C4-185500 and C4-185499 | 0.1.0 |
| 2018-08 | CT4#86 | C4-186339 |  |  |  | Implementation of C4-186508 and C4-186155 agreed in CT4#86 | 0.2.0 |
| 2018-10 | CT4#86bis | C4-187646 |  |  |  | Implementation of C4-187530, C4-187531, C4-187602, C4-187533, C4-187603, C4-187535 and C4-187536 agreed in CT4#86bis. | 0.3.0 |
| 2018-12 | CT4#87 | C4-188698 |  |  |  | Implementation of C4-188588 and C4-188589 agreed in CT4#87 | 0.4.0 |
| 2019-03 | CT4#89 | C4-190635 |  |  |  | Implementation of C4-190595, C4-190511, C4-190512, C4-190513, C4-190514, C4-190515, C4-190516, C4-190204 and C4-190205 agreed in CT4#89. | 0.5.0 |
| 2019-03 | CT#83 | CP-190050 |  |  |  | Presented for information | 1.0.0 |
| 2019-04 | CT4#90 | C4-191345 |  |  |  | General Cleanup | 1.1.0 |
| 2019-06 | CT4#91 | C4-192375 |  |  |  | QUIC impacts on transport proxy traversal | 1.2.0 |
| 2019-06 | CT4#91 | C4-192376 |  |  |  | Impacts of QUIC on Troubleshooting | 1.2.0 |
| 2020-03 | CT4#96E | C4-201161 |  |  |  | Additional requirements for Transport Protocol | 1.3.0 |
| 2020-03 | CT4#96E | C4-201164 |  |  |  | Update of QUIC features | 1.3.0 |
| 2020-06 | CT4#98E | C4-203159 |  |  |  | Interim Conclusion on HTTP/3 (QUIC) | 1.4.0 |
| 2020-06 | CT4#98E | C4-203532 |  |  |  | Architectural baseline requirements update | 1.4.0 |
| 2020-06 | CT4#98E | C4-203577 |  |  |  | MASQUE | 1.4.0 |
| 2020-09 | CT4#99E | C4-204161 |  |  |  | Update of the references | 1.5.0 |
| 2020-11 | CT4#101E | C4-205074 |  |  |  | Miscellaneous corrections | 1.6.0 |
| 2021-02 | CT4#101bis-E | C4-210157 |  |  |  | Clarifications to the basic HTTP/3 notions | 1.7.0 |
| 2023-08 | CT4#117 | C4-233153 |  |  |  | HTTP/3 implementation maturity and conclusion update | 1.8.0 |
| 2023-08 | CT4#117 | C4-233479 |  |  |  | HTTP/2 vs. HTTP/3 testing findings and considerations | 1.8.0 |
| 2023-08 | CT4#117 | C4-233496 |  |  |  | QUIC IETF RFC references update | 1.8.0 |
| 2023-09 | CT#101 | CP-232024 |  |  |  | TR Presented for approval | 2.0.0 |
| 2023-09 | CT#101 |  |  |  |  | TR Approved | 18.0.0 |