**3GPP TSG- Meeting #**

**, , -**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *CR-Form-v12.3* | | | | | | | | |
| **CHANGE REQUEST** | | | | | | | | |
|  | | | | | | | | |
|  |  | **CR** |  | **rev** |  | **Current version:** |  |  |
|  | | | | | | | | |
| *For* [***HE******LP***](http://www.3gpp.org/3G_Specs/CRs.htm#_blank)*on using this form: comprehensive instructions can be found at* [*http://www.3gpp.org/Change-Requests*](http://www.3gpp.org/Change-Requests)*.* | | | | | | | | |
|  | | | | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Proposed change affects:*** | UICC apps |  | ME | **x** | Radio Access Network |  | Core Network | **X** |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | |
| ***Title:*** |  | | | | | | | | | |
|  |  | | | | | | | | | |
| ***Source to WG:*** |  | | | | | | | | | |
| ***Source to TSG:*** | S4 | | | | | | | | | |
|  |  | | | | | | | | | |
| ***Work item code:*** |  | | | | |  | ***Date:*** | | |  |
|  |  | | | |  | |  | | |  |
| ***Category:*** |  |  | | | | | ***Release:*** | | |  |
|  | *Use one of the following categories:* ***F*** *(correction)* ***A*** *(mirror corresponding to a change in an earlier release)* ***B*** *(addition of feature),* ***C*** *(functional modification of feature)* ***D*** *(editorial modification)*  Detailed explanations of the above categories can be found in 3GPP [TR 21.900](http://www.3gpp.org/ftp/Specs/html-info/21900.htm). | | | | | | | | *Use one of the following releases: Rel-8 (Release 8) Rel-9 (Release 9) Rel-10 (Release 10) Rel-11 (Release 11) … Rel-17 (Release 17) Rel-18 (Release 18) Rel-19 (Release 19)  Rel-20 (Release 20)* | |
|  |  | | | | | | | | | |
| ***Reason for change:*** | | FS\_AMD includes the topic about opportunities with QUIC for segmented streaming (topic “m)”). The corresponding new clause “QUIC-based segmented media delivery” needs to be developped. | | | | | | | | |
|  | |  | | | | | | | | |
| ***Summary of change:*** | | Compared to the previous CR revision with TDoc n. S4-242125, this one integrates the changes from the agreed TDocs S4-242032 (Clause 5.24.7) and S4-242123 (Clause 5.24.1.2 and 5.24.6). | | | | | | | | |
|  | |  | | | | | | | | |
| ***Consequences if not approved:*** | |  | | | | | | | | |
|  | |  | | | | | | | | |
| ***Clauses affected:*** | | 5.24 [New] | | | | | | | | |
|  | |  | | | | | | | | |
|  | | **Y** | **N** |  | | | |  | | |
| ***Other specs*** | |  | **x** | Other core specifications | | | | TS/TR ... CR ... | | |
| ***affected:*** | |  | **x** | Test specifications | | | | TS/TR ... CR ... | | |
| ***(show related CRs)*** | |  | **x** | O&M Specifications | | | | TS/TR ... CR ... | | |
|  | |  | | | | | | | | |
| ***Other comments:*** | |  | | | | | | | | |
|  | |  | | | | | | | | |
| ***This CR's revision history:*** | | * CR-19rev2: [N/A] * CR-19rev3: Populate clause 5.24 * CR-19rev4: Update clause 5.24.1 and Figures * CR-19rev5: Added high-level protocol stack Figure * CR-19rev6: Updated Figure based on discussions * CR-19rev7: Integrates changes from agreed TDocs | | | | | | | | |

Change #1

## 2 References

…

[2] Akamai Blog, “A QUICk Introduction to HTTP/3”, April 2020, <https://www.akamai.com/blog/developers/a-quick-introduction-http3>

…

[5] IETF IETF RFC 9114: "HTTP/3", June 2022.

…

[W3C-WT] W3C Working Draft: "WebTransport", May 2024, <https://www.w3.org/TR/webtransport>

[WT-H3] A. Frindell, E. Kinnear, V. Vasiliev: "WebTransport over HTTP/3", Work In Progress, Internet Draft, draft-ietf-webtrans-http3, <https://datatracker.ietf.org/doc/html/draft-ietf-webtrans-http3/>

[HTTP-DGRAM] IETF RFC 9297: "HTTP Datagrams and the Capsule Protocol", August 2022.

[QUIC-DGRAM] IETF RFC 9221: "An Unreliable Datagram Extension to QUIC", March 2022.

[HTTP-PRIO] IETF RFC 9218: "Extensible Prioritization Scheme for HTTP", June 2022.

[DASH6] ISO/IEC 23009-6: "Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 6: DASH with server push and WebSockets".

[MoQ] L. Curly, K. Pugin, S. Nandakumar, V. Vasiliev, I. Swett: "Media over QUIC Transport", Work In Progress, Internet Draft, draft-ietf-moq-transport, https://datatracker.ietf.org/doc/draft-ietf-moq-transport/07/

[MOQ-CCF] S. Nandakumar, W. Law, M. Zanaty: "Common Catalog Format for moq-transport", Work In Progress, Internet Draft, draft-ietf-moq-catalogformat, https://datatracker.ietf.org/doc/draft-ietf-moq-catalogformat/01/

[x6] Zhang, X., Jin, S., He, Y., Hassan, A., Mao, Z. M., Qian, F., & Zhang, Z. L., "QUIC is not Quick Enough over Fast Internet", in *Proceedings of the ACM on Web Conference 2024* (pp. 2713–2722), May 2024.

[x7] Nguyen, M., Nys, P., Pham, S., Silhavy, D., Arbanowski, S., & Steglich, S., "Toward WebTransport Support in HTTP Adaptive Streaming", in *Tenth International Conference on Communications and Electronics 2024* (ICCE) (pp. 96–101), IEEE, July 2024.

Change #2  
(all new Text)

## 5.24 QUIC-based segmented media delivery

### 5.24.1 Description

#### 5.24.1.1 General

QUIC, specified in RFC 9000 [32], is a secure, reliable, multiplexed, connection-oriented transport protocol built on top of UDP. It is widely available and its impact on media streaming requires further study. In this clause QUIC-specific media streaming is studied, not necessarily based on HTTP/3 [5] that is studied in clause 5.4.

A QUIC client establishes a connection with a server, and within this connection multiple concurrent streams can transport data. Thanks to a more efficient implementation of the TLS initial handshake, a QUIC connection is typically established faster than a TCP + TLS connection, therefore reducing initialization time. Additionally, by allowing QUIC streams to be multiplexed into a single QUIC connection, they can operate independently of each other, each with its own separate congestion window. Because packet retransmission occurs within a stream, a stall in one stream does not block the progress in the other streams. Similarly, packet loss in one stream does not affect the progress of data transfer in other streams multiplexed in the same QUIC connection. Finally, QUIC, like HTTP/2 [4], enables a sender to prioritise traffic at an individual stream level. Unlike HTTP/2, however, QUIC does not provide the means to signal this prioritisation to its connection peer, and so it is entirely up to the sending application to set appropriate priorities when multiplexing streams.

Even though QUIC solves many issues compared to TCP e.g. faster connection establishing and non-blocking multiplexing (using streams), there are some open issues and shortcomings when it comes to media content delivery.

A 5GMS Application Provider runs an adaptive media streaming service between a 5GMS AS and a 5GMS Client running on a UE using 5G Media Streaming protocols conveyed at reference points M2 and M4. However, only M4 is relevant for this key topic since it focuses on the media delivery to the UE and not on the ingest of the media itself. Also, since M5 is not meant to transport media, this reference point is also excluded from this key topic.

#### 5.24.1.2 Relevant existing technologies

##### 5.24.1.2.1 Extensible Prioritization Scheme for HTTP

RFC 9218 [HTTP-PRIO] defines the signalling for an HTTP client to express its preferences in terms of relative priorities when multiple requests are concurrently sent. Upon reception, the HTTP server can make use of this information for serving the response deemed to be of highest priority by the client. Similarly, an HTTP server may use this prioritisation scheme to inform downstream entities about the relative priorities of multiple responses.

##### 5.24.1.2.2 MPEG-DASH Part 6: DASH with Server Push and WebSockets

Published in 2017, MPEG-DASH Part 6 [DASH-6] is the part of the MPEG-DASH standard that enables a DASH server to send multiple segments to a DASH client without the need to receive individual segment requests. To this end, [DASH-6] defines the concept of push directives sent by the DASH client to the DASH server, indicating which segments should be pushed. Two alternative instantiations are defined in [DASH-6]:

1. Based on HTTP/2 [4] server push functionality with specific HTTP header extensions for the DASH client to send the so-called push directives.

2. Based on the WebSocket protocol, and comprising a set of messages for the exchange of MPD, segment and push directives. This constitutes the WebSocket sub-protocol for DASH registered with IANA as 2016.serverpush.dash.mpeg.org. The WebSocket sub-protocol for DASH uses a binary format for all messages exchanged over the WebSocket connection.

##### 5.24.1.2.3 WebTransport

The WebTransport protocol developed by the IETF can be layered over either HTTP/2 [4] or HTTP/3 [5].

When it is layered over HTTP/3 [WT-H3], WebTransport allows a client and a server to communicate over a secure, multiplexed QUIC-based transport. WebTransport leverages the QUIC streams to offer unidirectional and bidirectional streams and guarantee reliable, ordered reception of byte streams. WebTransport over HTTP/3 relies on HTTP/3 to establish the WebTransport session with the session peer using the extended CONNECT HTTP method. After that point, the application directly integrates with the QUIC layer.

##### 5.24.1.2.4 Push-based adaptive media streaming over WebTransport with server-side throughput estimation

The conference paper [x7] proposes a mechanism to support HTTP adaptive media streaming in the WebTransport protocol. In essence, this works as follows:

1. The delivery of the MPD to the DASH client is assumed to have taken place prior to the establishment of the WebTransport transport connection for media.

2. The WebTransport-capable DASH client connects to a WebTransport-capable DASH server and establishes a WebTransport connection.

3. The DASH server sends media segments of default start-up quality for the audio and video Adaptation Setsto the DASH client.

NOTE: In the prototype implementation described in [x7], there is only one audio Adaptation Set listed in the DASH MPD and only one video Adaptation Set, so there is no need for the DASH client to indicate which ones it is interested in receiving.

4. The DASH client starts receiving media segments and starts sending back qlog metrics reports [84] (as summarised in clause 5.4.1.6) to the DASH server on the same bidirectional stream.

5. Based on the received qlog metrics reports, the DASH server continuously estimates the throughput available to the DASH client.

6. When the new throughput is estimated, the DASH server may decide to change the pushed Representation or inform the DASH client about the new estimated throughput for the DASH client to switch. (The paper [x7] does not explicitly mention which option is used.)

##### 5.24.1.2.4 Media-over-QUIC Transport

The Media-over-QUIC transport protocol [MoQ] is based on QUIC or can alternatively be layered on top of WebTransport. Although generic, the transport protocol has been initially designed for media delivery. Based on a publish–subscribe interaction pattern, it allows a publisher to distribute media content to many subscribers with a focus on latency and scalability. Since QUIC and WebTransport are merely opening communication channels between client and server, MoQT defines a set of messages to establish and operate a MoQT session. In particular, a MoQT client can indicate the desired media content to receive by sending a SUBSCRIBE message to a publisher.

#### 5.24.1.3 Application access to QUIC protocol features

In a common scenario, QUIC is not directly exposed to the application, but it is more typically mediated through a QUIC-enabled application protocol such as HTTP/3 as specified in RFC 9114 [5] or Media over QUIC Transport [MoQ], or else through a protocol framework such as WebTransport [W3C-WT] mapped onto HTTP/3 [WT-H3]. For this reason, the set of QUIC features available to an application is the subset exposed by the chosen application protocol or framework (typically invoked via the public API of a client or server library). This approach facilitates the development of QUIC-based communication but at the cost of limiting the level of control an application has over the connection and stream management.

#### 5.24.1.4 Connection and stream management

Applications are encouraged to keep QUIC connections alive when it makes sense to do so, and HTTP/3 client libraries typically facilitate connection keep-alive behaviour for efficiency reasons. In addition, QUIC’s "0-RTT" connection establishment procedure allows a client to reconnect to a server it has previously connected to, and to reuse a security context negotiated during a previous connection and cached by both parties to send application payload data to the server in the first UDP datagram of the new QUIC connection.

In the context of HTTP/3 [5], the mechanism for handling streams is tightly specified by the HTTP/3 protocol that is each HTTP request-response transaction consumes one stream in the QUIC connection.

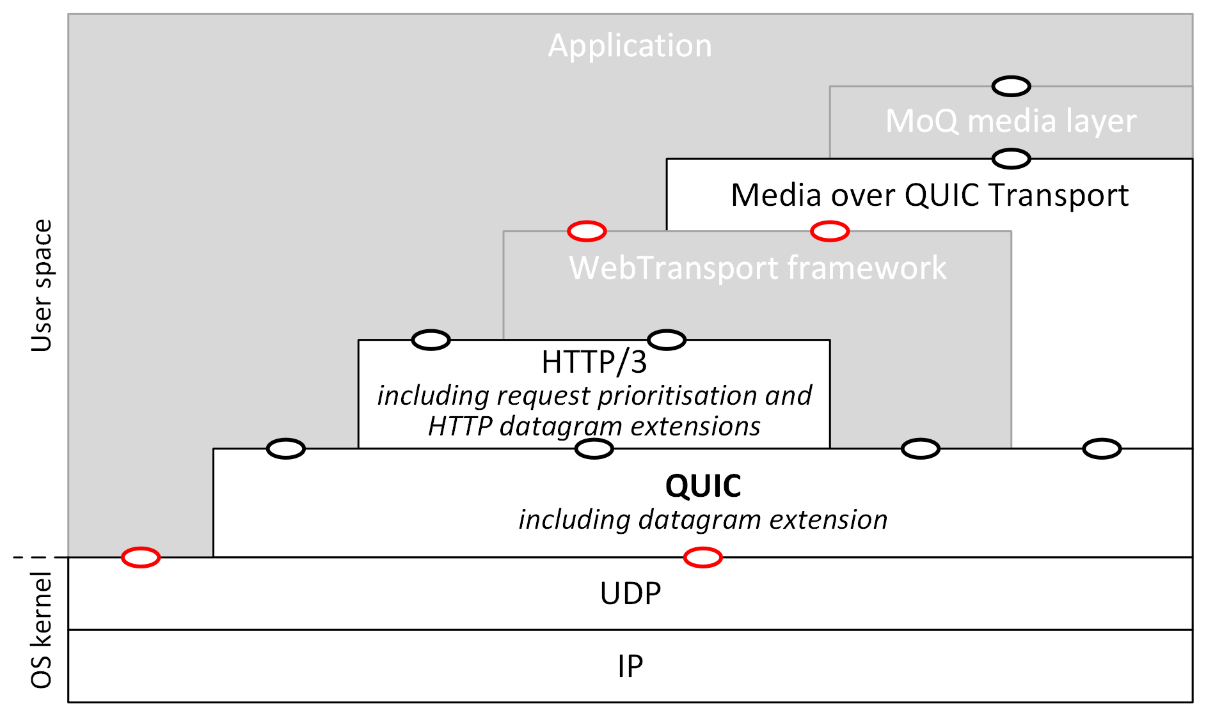
In contrast, WebTransport [W3C-WT] allows the application a greater degree of control over QUIC connections and streams when layered on top of HTTP/3 [WT-H3]. However, the use of WebTransport for segmented media delivery is not a well-studied area. In one implementation of WebTransport for segmented media streaming [x7] the client sends metrics to the server, the server performs throughput estimation and then sends the respective segments to the client.

QUIC allows both the client and the server to initiate the opening of a new stream in a connection and the WebTransport API exposes this capability to applications. This feature is not directly supported by HTTP/3 [5] where a server push is always stimulated by a client-initiated request.

The protocol extension in RFC 9221 [QUIC-DGRAM] specifies how peers in a QUIC connection can exchange unreliable application messages using a special-purpose QUIC datagram frame. This type of frame is acknowledged by the QUIC recipient, but is never retransmitted if left unacknowledged. Section 2 of RFC 9297 [HTTP-DGRAM] specifies how QUIC datagrams can be used to support the exchange of unreliable HTTP messages in HTTP/3. Furthermore, [WT-H3] specifies how this feature can be exposed in the WebTransport framework when that is layered on top of HTTP/3 (and hence on top of QUIC).

Finally, QUIC supports different congestion control algorithms and control over this is exposed to the application in the WebTransport API [W3C-WT].

An overview of the protocol stack is illustrated in figure 5.24.1.1-1. Everything in user space is potentially accessible from the application and must rely on the implementation of the API, while the layers in kernel space are accessed via Operating System calls (e.g. UDP can typically only be accessed via an API provided by the OS-provided socket library).



NOTE 1: White boxes denote protocol layers; grey boxes non-protocol layers, such as client/server frameworks.

NOTE 2: Ovals denote instantiations of the API defined by a given block, with red colour indicating a standardized API and black indicating non-standardized implementation-specific APIs (e.g. library APIs).

NOTE 3: The WebTransport framework intiallly uses HTTP/3 for connection establishment and afterwards switches to interfacing with the QUIC layer for data transfer.

NOTE 4: The Media over QUIC Transport layer [MoQ] typically uses QUIC directly, but can also be layered on top of WebTransport.

NOTE 5: The MoQ media layer uses the MoQ Streaming Formats catalogue [MOQ-CCF] that defines policies on content discovery and subscription and information on encoding, packaging and mapping the content to MoQ objects.

Figure 5.24.1.1-1: QUIC high-level protocol stack

#### 5.24.1.5 Stream prioritisation

By design, QUIC does not provide the means for connection peers to signal the priority of a stream, leaving decision-making over stream prioritisation to the application. To support this, section 2.3 of RFC 9000 [32] recommends that QUIC implementations should expose the means for applications to set relative stream priorities, thus providing the ability to influence the scheduling of packets for transmission in a QUIC connection. Thus, stream prioritisation is typically handled on the higher application layers, but neither the network nor the recipient is aware of the applied mechanism. Relevant standardization efforts include an extensible prioritisation scheme for HTTP (including HTTP/3) defined in RFC 9218 [HTTP-PRIO], and prioritisation scheme for the Media over QUIC (MoQ) transport protocol [MoQ] currently being developed by IETF.

#### 5.24.1.6 Computational overhead of QUIC endpoints

Adoption of QUIC-based application protocols such as HTTP/3 [5], Media over QUIC Transport [MoQ] and application frameworks such as WebTransport [W3C-WT] can have impact on the UEs due to the processing required for the underlying QUIC protocol itself. In TCP, most functions are executed in kernel space, which over the years have been heavily optimised including, in some hosts, hardware offload. In QUIC, by contrast, most protocol functions are executed in user space.

There are two identified sources for this computational overhead [x6]:

1. The in-kernel UDP stack issues many packet reads; this is because each datagram arriving in the link layer is forwarded for processing to the transport layer i.e., there are no offload mechanisms used like the sender-side Generic Segmentation Offload (GSO), or the generic receive offload mechanism (GRO) used by the link layer module to combine datagrams into a mega datagram before forwarding it to the transport layer.

2. There is increased overhead of processing packets (and, as a result, in generating responses) due to increased user space processing required as a result of 1 above (i.e. all the packets need to be processed individually) combined with managing QUIC acknowledgements in user space (instead of kernel space as with TCP in HTTP/2).

These potential performance effects should be taken in consideration because they can negatively impact the overall Quality of Service that can be obtained from a QUIC connection, and can increase the energy consumption of the UE.

#### 5.24.1.7 Key Issue objectives

The objectives listed below are targeting QUIC-specific delivery aspects:

• Application access to QUIC connections (e.g. via QUIC libraries).

• Variability of QUIC implementations.

### 5.24.2 Collaboration scenarios

#### 5.24.2.1 General

For the purpose of describing the following scenarios, it is only assumed that the 5GMS Client supports the QUIC protocol. Additionally, the 5GMS Client may support higher level protocols based on QUIC (for instance HTTP/3 or WebTransport) but this is not required, and the analysis of those collaboration scenarios is still applicable to any of those cases.

#### 5.24.2.2 QUIC-agnostic 5GMS Client

In this scenario, the Media Stream Handler of the 5GMS Client operates a QUIC session over reference point M4 but the 5GMS Client has no specific feature regarding QUIC. This has the advantage that 5GMS Client is generic and implements the same logic whether or not QUIC is used for the delivery of the media.

A diagram of a software application

Description automatically generated with medium confidence

Figure 5.24.2.2-1: QUIC-agnostic 5GMS Client

#### 5.24.2.3 Media-independent QUIC-aware 5GMS Client

In this scenario, the Media Stream Handler of the 5GMS Client operates a QUIC session over reference point M4 and the 5GMS Client is able to detect whether QUIC is used and, in case it is used, the 5GMS Client can apply different logic.

A diagram of a software application

Description automatically generated with medium confidence

Figure 5.24.2.3-1: Media-independent QUIC-aware 5GMS Client

In this case, the QUIC client implementation is not specifically optimised for media transport (e.g. a generic off-the-shelf QUIC client library) and the set of QUIC protocol features exposed to the Media Stream Handler is limited by the richness of its API.

With some limited control over the QUIC streams, such a 5GMS Client would typically be able to:

* Set relative priorities between the different QUIC streams. Relative stream priorities can be useful to differentiate audio and video, base layer and enhancement layer, etc.
* Receive updates sent proactively by the 5GMS AS. For example, a 5GMSd AS could send MPD updates to a Media Player using this mechanism.

#### 5.24.2.4 Media-optimised QUIC-aware 5GMS Client

In this scenario, the Media Stream Handler of the 5GMS Client operates a QUIC session over reference point M4 and the 5GMS Client is able to control the delivery of the media within the QUIC session.

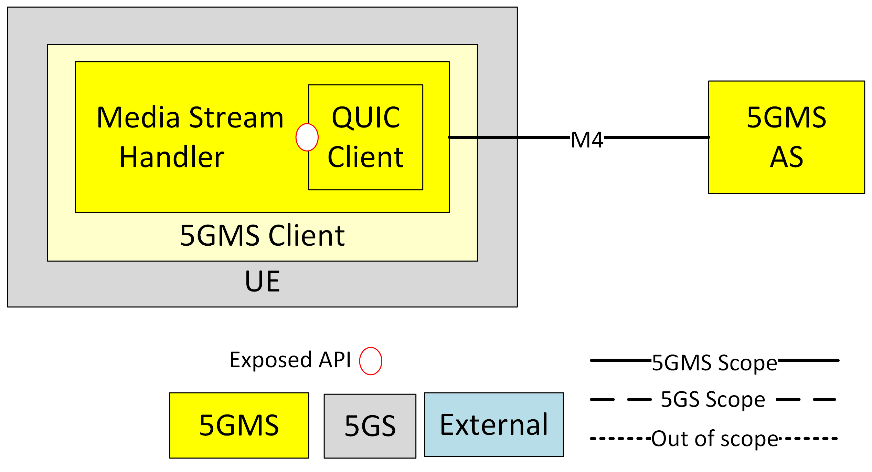


Figure 5.24.2.4-1: Media-optimised QUIC-aware 5GMS Client

In this case, the QUIC client implementation is optimised for media transport and the set of QUIC protocol features exposed to the Media Stream Handler is therefore unlimited. Hence, the media-optimised, QUIC-aware 5GMS Client provides the finest control over the delivery of media within the QUIC session.

With fine control over the QUIC streams, such a 5GMS Client would typically be able to:

* Set relative priorities between the different QUIC streams. Relative stream priorities can be useful to differentiate audio and video, base layer and enhancement layer, etc.
* Receive updates sent proactively by the 5GMS AS. For example, a 5GMSd AS could send MPD updates to a Media Player using this mechanism.
* Use one QUIC stream for all the media segments of a given component (e.g. per CMAF Track).

### 5.24.3 Architecture mapping

#### 5.24.3.1 General

In the 5GMS architecture, the Media Stream Handler in the 5GMS Client is connected to the 5GMS Application Server (5GMS AS) via reference point M4. Therefore, both the client and the server on that interface need to be compatible in terms of protocols and functionalities. In the clause, the QUIC Client module which is part of the 5GMS Client will thus be connected to a QUIC Server connected or part of the 5GMS AS. Three mappings are considered, one for each collaboration scenario.

#### 5.24.3.2 Mapping with a QUIC-agnostic 5GMS Client

In this mapping, both 5GMS Client and 5GMS AS are agnostic to the QUIC protocol and the usage of QUIC as a network protocol is thus transparent for system. The integration of the QUIC Client in the UE and the QUIC Server with the 5GMS AS is considered to be out of scope of the 5GMS System.

A close-up of a computer screen

Description automatically generated

Figure 5.24.3.2-1: Architecture with QUIC-agnostic 5GMS Client and 5GMS AS

#### 5.24.3.3 Mapping with a Media-independent QUIC-aware 5GMS Client

In this mapping, the 5GMS Client and the 5GMS AS are integrated with, respectively, a QUIC Client and QUIC Server. The QUIC Client and the QUIC Server are external to the systems which means that any software implementation may be used. The integrations of the QUIC Client in the UE and the QUIC Server with the 5GMS AS are achieved via the exposed APIs by both the QUIC Client and QUIC Server. However, since the QUIC Client and Server are not a standardised part of the 5GMS System, the APIs exposed by the QUIC Client and the QUIC Server may thus also differ in capabilities.

In addition, the QUIC Client and the QUIC Server need to support a common QUIC protocol version. The QUIC protocol does not provide the means for negotiating the protocol version when establishing the connection, but rather enables a QUIC Server to reject unsupported version and propose alternative versions to a QUIC Client. This step of version selection may be influenced by Media Stream Handler and the 5GMS AS through those APIs.

A close-up of a computer screen

Description automatically generated

Figure 5.24.3.3-1: Architecture with QUIC-aware 5GMS Client and 5GMS AS

#### 5.24.3.4 Mapping with a Media-optimised QUIC-aware 5GMS Client

In this mapping, the 5GMS Client and the 5GMS AS are integrated with, respectively, a QUIC Client and QUIC Server. The QUIC Client and the QUIC Server are part of the 5GMS System which means that their software implementation follows requirement and functionalities supporting the 5GMS Client’s needs. The integrations of the QUIC Client with the Media Stream Handler in the UE and the QUIC Server with the 5GMS AS are achieved via the exposed APIs by both the QUIC Client and QUIC Server. Since the functionality of the QUIC Client and Server are standardised as part of the 5GMS System in this mapping, the APIs exposed by the QUIC Client and the QUIC Server are compatible in terms of functionalities. In addition, the QUIC Client and the QUIC Server support a commonly agreed QUIC version to guarantee the establishment of the QUIC connection. In addition, those client and server APIs enables the 5GMS Client to optimise the media delivery.

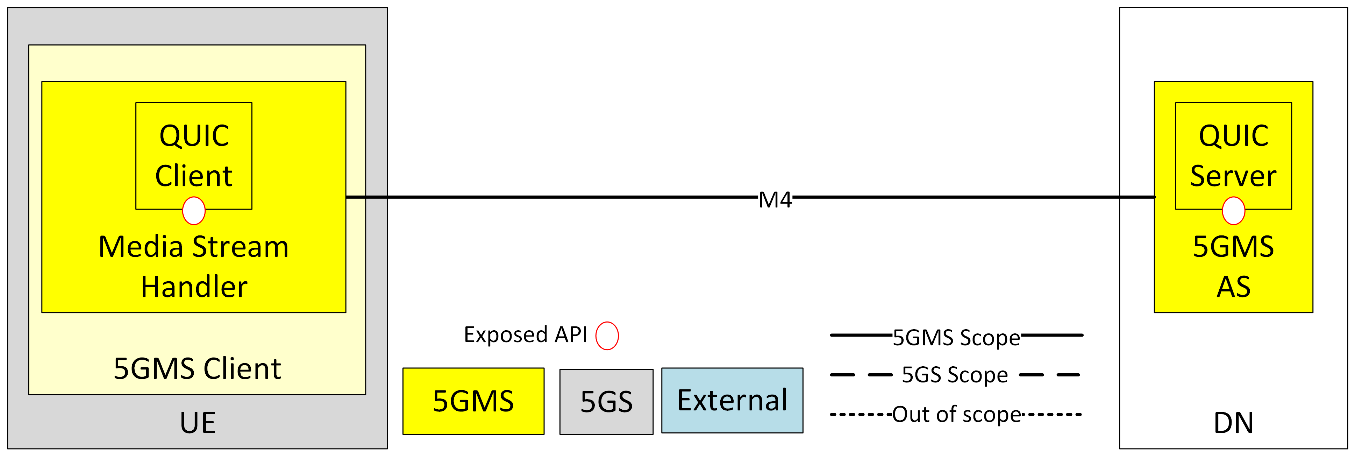


Figure 5.24.3.4-1: Architecture with media-optimised QUIC-aware 5GMS Client and 5GMS AS

### 5.24.4 High-level call flows

#### 5.24.4.1 General

Editor’s Note: Further content to be provided.

### 5.24.5 Gap analysis and requirements

An analysis of the gaps identified in clause 5.24.4 and the derivation of requirements from these are for further study.

### 5.24.6 Candidate solutions

#### 5.24.6.1 General

The candidate solutions in the following clauses are considered in relation to instantiation of the following types of 5GMS Client:

- Media-independent QUIC-aware 5GMS Client, as introduced in clause 5.24.2.3 and as mapped in clause 5.24.3.3.

- Media-optimised QUIC-aware 5GMS Client, as introduced in clause 5.2.4.2.4 and as mapped in clause 5.24.3.4.

NOTE: The QUIC-agnostic 5GMS Client type is covered by clause 5.4.

Additional candidate solutions may be identified subsequently, subject to further study.

#### 5.24.6.2 MPEG-DASH over HTTP/3 with server push and priority information

This candidate solution is an instantiation of the media-independent QUIC-aware 5GMS Client based on the following technologies:

- MPEG-DASH Part 1 [11].

- The HTTP/3 protocol as specified in RFC 9114 [5], including the server push functionality.

- The Extensible Prioritization Scheme for HTTP as specified in RFC 9218 [HTTP-PRIO].

This candidate solution enables the delivery of DASH content to a 5GMSd Client using the HTTP/3 protocol at reference point M4d. In addition, both the 5GMSd AS and the Media Player support the server push functionality and the extensible prioritisation scheme for HTTP.

The call flow in clause 5.24.4.3 applies.

#### 5.24.6.3 MPEG-DASH Part 6 over WebTransport

##### 5.24.6.3.1 Prerequisites

This candidate solution is an instantiation of the media-optimised QUIC-aware 5GMS Client based on the following technologies:

- MPEG-DASH Part 1 [11].

- MPEG-DASH Part 6 [DASH-6].

- WebTransport protocol over HTTP/3 [WT-H3].

This candidate solution enables the delivery of DASH content to a UE using the WebTransport protocol layered over HTTP/3 at reference point M4. In addition, both the 5GMSd AS and the Media Player support the WebSocket sub-protocol for DASH specified in clause 8.2 of MPEG-DASH Part 6 [DASH-6]. However, the communication runs over a WebTransport connection instead of a WebSocket connection.

The call flow in clause 5.24.4.4 applies.

#### 5.24.6.4 Media-over-QUIC

This candidate solution is an instantiation of the push-based QUIC-aware 5GMS Client based on the following technologies:

- MPEG-DASH Part 1 [11].

- Media-over-QUIC Transport protocol [MoQ]

This candidate solution enables the delivery of DASH content to a UE using the WebTransport protocol over HTTP/3 or the QUIC protocol at reference point M4. In addition, both the 5GMSd AS and the Media Player support the Media-over-QUIC Transport protocol. The media content delivered using the Media-over-QUIC Transport protocol is assumed to be DASH segments, and it is also assumed that the MPD has been delivered in the initialisation phase such that the Media Player is able to generate SUBSCRIBE messages with the information related to desired Representation or simply called tracks in the context of MoQ.

The call flow in clause 5.24.4.5 applies.

#### 5.24.6.5 Push-based adaptive media streaming over WebTransport with server-side throughput estimation

This candidate solution is an instantiation of the push-based QUIC-aware 5GMS Client based on the following technologies:

- MPEG-DASH Part 1 [11].

- WebTransport protocol over HTTP/3 [WT-H3].

- A specific WebTransport sub-protocol for MPEG-DASH as specified in [x7].

This candidate solution enables the delivery of DASH content to a UE using the WebTransport over HTTP/3 protocol at reference point M4. In addition, both the 5GMSd AS and the Media Player support the protocol described in [x7] (as summarised in clause 5.24.1.2.3).

The call flow in clause 5.24.4.5 applies.

### 5.24.7 Summary and conclusions

The study of this Key Issue has explored the ways in which QUIC can enhance the deployment of 5G Media Streaming services, and the potential open issues arising from this deployment. Based on the candidate solutions and the overall capabilities of QUIC, the following conclusions can be drawn.

QUIC is a widely deployed and supported protocol. However, when it comes to using QUIC for media delivery, the early deployments of segmented streaming over QUIC were achieved by using the HTTP/3 protocol for MPD and segment requests, as documented in clause 5.4. This approach is QUIC-agnostic and thus does not take full advantage of the capabilities offered by QUIC. In some cases, as reported by clause 5.4, there may even be some degradations of the performances compared to using HTTP/1.1. When it comes to non-QUIC-agnostic approaches, there exist several candidate technologies ranging from early standardization efforts based on QUIC, previous standardization activities based on HTTP/2 and WebSocket that could be reused with QUIC and WebTransport, as well as research prototypes natively built on QUIC and WebTransport. As a result, it appears that there is thus no obvious path, from the candidate solutions, for enhancing segmented delivery by leveraging the emergence of the QUIC protocol.

At this stage, there is thus no basis justifying the need for any normative work.

However, the following aspects are to be further studied:

- The fragmentation of QUIC implementations (both client and server side) and their performances for segmented media delivery.

- The impact of reporting Qlog metrics via the downlink interface (M4) to the UE (bidirectional streams).

- The standardized definitions and their availability on the market of QUIC client and server APIs.

- The standardized definitions and their availability on the market of WebTransport client and server APIs.

- Evaluation of the identified candidate technologies with regards to QoE metrics (start-up delay, stalling events, etc.).

END OF CHANGES