JENAM TECH, LLC'S FIRST AMENDED SET OF INFRINGEMENT CONTENTIONS

U.S. Patent No. 10,951,742 – Google LLC
Claims 1-3, 6, 10, 12, 15, 20-21, 27-28, 35, 39, 48, 51-52, 55-56, 63-65, 67-69, 71-78, 80, 83-84, 86, 88, 103-104, 159, 167-176

Jenam Tech LLC ("Jenam") provides evidence of infringement of claims 1-3, 6, 10, 12, 15, 20-21, 27-28, 35, 39, 48, 51-52, 55-56, 63-65, 67-69, 71-78, 80, 83-84, 86, 88, 103-104, 159, 167-176 of U.S. Patent No. 10,951,742 (hereinafter "the '742 patent") by Google LLC ("Google"). In support thereof, Jenam provides the following claim charts.

"Accused Instrumentalities" as used herein refers to at least one or more websites or web addresses including, but not limited to www.google.com, stored and/or hosted on one or more servers owned or under the control of Google. For the sake of clarity, Jenam alleges that all of Google's various websites, products, and platforms utilizing QUIC infringe the '742 patent, including, but not limited to Google Edge Network, Google Cloud, Chrome Enterprise, G suite, Google Play, Chrome, Android (Android Enterprise, Android Messages (RCS)), Duo, Google Ads, Adwords, Google Analytics, Youtube, Google Mobile apps, Google shopping, and Google Maps. A list of Google's "products" can be found at https://about.google/intl/en_us/products/. On information and belief, the Accused Instrumentalities referred to in the charts below are representative of Google's use of QUIC in Google's other websites, products and platforms. These claim charts demonstrate Google's infringement, and provide notice of such infringement, by comparing each element of the asserted claims to corresponding components, aspects, and/or features of the Accused Instrumentalities. These claim charts are not intended to constitute an expert report on infringement. These claim charts include information provided by way of example, and not by way of limitation.

The analysis set forth below is based only upon information from publicly available resources regarding the Infringing Instrumentalities. An analysis of Google's (or other third parties') technical documentation and/or software source code may assist in fully identify all infringing features and functionality. Accordingly, Jenam reserves the right to supplement this infringement analysis once such information is made available to Jenam. Furthermore, Jenam reserves the right to revise this infringement analysis, as appropriate, upon issuance of a court order construing any terms recited in the asserted claims.

Unless otherwise noted, Jenam contends that Google directly infringes the '742 patent in violation of 35 U.S.C. § 271(a) by selling, offering to sell, making, using, and/or importing the Infringing Instrumentalities. The following exemplary analysis demonstrates that infringement. Unless otherwise noted, Jenam further contends that the evidence below supports a finding of indirect infringement under 35 U.S.C. §§ 271(b) and/or (c), in conjunction with other evidence of liability under one or more of those subsections. Google makes, uses, sells, imports, or offers for sale in the United States, or has made, used, sold, imported, or offered for sale in the past, without authority, or induces others to make, use, sell, import, or offer for sale in the United States, or has induced others to make, use, sell, import, or offer for sale in the past, without authority products, equipment, or services that infringe claims 1-3, 6, 10, 12, 15, 20-21, 27-28, 35, 39, 48, 51-52, 55-56, 63-65, 67-69, 71-78, 80, 83-84, 86, 88, 103-104, 159, 167-176 of the '742 patent, including without limitation, the Accused Instrumentalities.

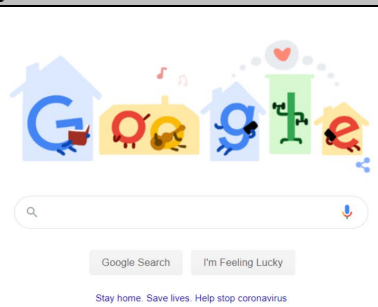
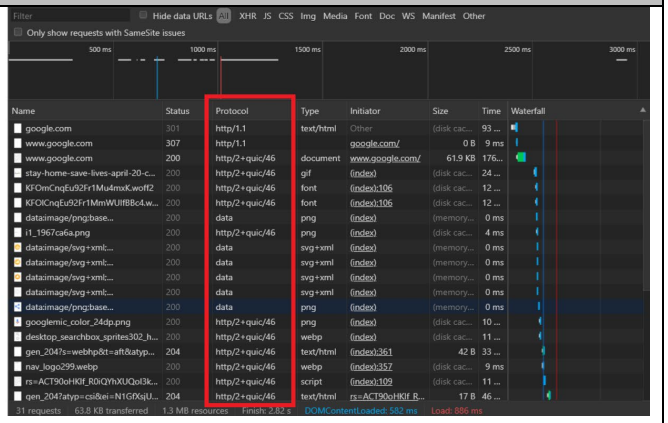
Unless otherwise noted, Jenam believes and contends that each element of each claim asserted herein is literally met through Google's provision of the Infringing Instrumentalities. However, to the extent that Google attempts to allege that any asserted claim element is not literally met, Jenam believes and contends that such elements are met under the doctrine of equivalents. More specifically, in its investigation and analysis of the Infringing Instrumentalities, Jenam did not identify any substantial differences between the elements of the patent claims and the corresponding features of the Infringing Instrumentalities, as set forth herein. In each instance, the identified feature of the Infringing Instrumentalities performs at least substantially the same function in substantially the same way to achieve substantially the same result as the corresponding claim element.

To the extent the chart of an asserted claim relies on evidence about certain specifically-identified Accused Instrumentalities, Jenam asserts that, on information and belief, any similarly-functioning instrumentalities also infringes the charted claim. Jenam reserves the right to amend this infringement analysis based on other products made, used, sold, imported, or offered for sale by Google. Jenam also reserves the right to amend this infringement analysis by citing other claims of the '742 patent, not listed in the claim chart, that are infringed by the Accused Instrumentalities. Jenam further reserves the right to amend this infringement analysis by adding, subtracting, or otherwise modifying content in the "Accused Instrumentalities" column of each chart.

Claim	Claim Elements	Applicability
1	An apparatus, comprising: a non-transitory memory storing instructions; and one or more processors in communication with the non-transitory memory, wherein the one or more processors execute the instructions to:	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.), comprising: a non-transitory memory storing instructions (e.g., server software, etc.), and one or more processors in communication with the non-transitory memory, wherein the one or more processors execute the instructions to: See excerpt(s) below, for example (emphasis added, if any): **Note**: Google admits that their "Google Cloud Platform" uses QUIC: "Google Cloud Platform Blog... we're happy to be the first major public cloud to offer QUIC support for our HTTPS load balancers." https://cloudplatform.googleblog.com/2018/06/Introducing-QUIC-support-for-HTTPS-load-balancing.html

Claim	Claim Elements	Applicability
		Note: The "Google Cloud Platform" that, as established above, uses QUIC, is also used for ALL of Google's services: "Google Cloud Platform (GCP), offered by Google (company), is a suite of cloud computing services that runs on the same infrastructure that Google uses internally for its end-user products, such as Google Search, Gmail, file storage, and YouTube" https://en.wikipedia.org/wiki/Google_Cloud_Platform **Note**: At least a portion of the citations herein are made to the QUIC standard found at: https://tools.ietf.org/html/draft-ietf-quic-transport-22, and/or https://tools.ietf.org/html/draft-ietf-quic-transport-27. Based upon information and belief, the pertinent portions of such version of the QUIC standard may also be found in one or more other versions. For completeness, all of the versions below (and future iterations thereof) are hereby incorporated by reference: https://tools.ietf.org/html/draft-ietf-quic-transport-00 https://tools.ietf.org/html/draft-ietf-quic-transport-01 https://tools.ietf.org/html/draft-ietf-quic-transport-02 https://tools.ietf.org/html/draft-ietf-quic-transport-03 https://tools.ietf.org/html/draft-ietf-quic-transport-04 https://tools.ietf.org/html/draft-ietf-quic-transport-05 https://tools.ietf.org/html/draft-ietf-quic-transport-06 https://tools.ietf.org/html/draft-ietf-quic-transport-07 https://tools.ietf.org/html/draft-ietf-quic-transport-08 https://tools.ietf.org/html/draft-ietf-quic-transport-09 https://tools.ietf.org/html/draft-ietf-quic-transport-10 https://tools.ietf.org/html/draft-ietf-quic-transport-11 https://tools.ietf.org/html/draft-ietf-quic-transport-12 https://tools.ietf.org/html/draft-ietf-quic-transport-13 https://tools.ietf.org/html/draft-ietf-quic-transport-14 https://tools.ietf.org/html/draft-ietf-quic-transport-15

Claim	Claim Elements	Applicability
		https://tools.ietf.org/html/draft-ietf-quic-transport-16 https://tools.ietf.org/html/draft-ietf-quic-transport-17 https://tools.ietf.org/html/draft-ietf-quic-transport-18 https://tools.ietf.org/html/draft-ietf-quic-transport-19 https://tools.ietf.org/html/draft-ietf-quic-transport-20 https://tools.ietf.org/html/draft-ietf-quic-transport-21 https://tools.ietf.org/html/draft-ietf-quic-transport-22 https://tools.ietf.org/html/draft-ietf-quic-transport-23 https://tools.ietf.org/html/draft-ietf-quic-transport-24 https://tools.ietf.org/html/draft-ietf-quic-transport-25 https://tools.ietf.org/html/draft-ietf-quic-transport-26 https://tools.ietf.org/html/draft-ietf-quic-transport-27 **Note:** Though these contentions may refer to a specific version of the QUIC standard, the presented evidence is not exclusive to this specific version and is compliant with and/or supported by current versions of the QUIC standard, including QUIC_VERSION_IETF_RFC_V1 (https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-34), QUIC_VERSION_IETF_DRAFT_25 (https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-25), QUIC_VERSION_IETF_DRAFT_27 (https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-27), QUIC_VERSION_IETF_DRAFT_28 (https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-28), QUIC_VERSION_IETF_DRAFT_29 (https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-29), and/or QUIC_VERSION_50 which relies in part on IETF_QUIC_INVARIANTS_06 (https://tools.ietf.org/html/draft-ietf-quic-invariants-06) and/or IETF_QUIC_TRANSPORT_17 (https://tools.ietf.org/html/draft-ietf-quic-transport-17), each of which, upon information and belief, has been utilized by Google LLC through its accused instrumentalities. **Note**: Below is a web page of Google (https://www.google.com/).

Claim	Claim Elements	Applicability
		 
	identify, at a first node, first information on which at least a first duration for detecting a first type of time period is based;	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.), configured to identify, at a first node (e.g., one of a QUIC-compliant server or client, etc.), first information (e.g., first value, etc.) on which at least a first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.) for detecting a first type of time period (e.g., idle-type timeout, etc.) is based; See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily detected based on the value received in the idle_timeout field of the connection negotiation packet. "idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
		"10. Connection Termination An established QUIC connection can be terminated in one of three ways: • idle timeout (Section 10.2) • immediate close (Section 10.3) • stateless reset (Section 10.4) An endpoint MAY discard connection state if it does not have a validated path on which it can send packets (see Section 8.2)." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried."

Claim	Claim Elements	Applicability
		QUIC: A UDP-Based Multiplexed and Secure Transport https://tools.ietf.org/html/draft-ietf-quic-transport-22 "7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and active_connection_id_limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters.

Claim	Claim Elements	Applicability
		• initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial max data and either initial max streams bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "3.3. Session resumption versus Keep-alive

Claim	Claim Elements	Applicability
		Because QUIC is encapsulated in UDP, applications using QUIC must deal with short idle timeouts. Deployed stateful middleboxes will generally establish state for UDP flows on the first packet state, and keep state for much shorter idle periods than for TCP. According to a 2010 study ([Hatonen10]), UDP applications can assume that any NAT binding or other state entry will be expired after just thirty seconds of inactivity. A QUIC application has three strategies to deal with this issue: • Ignore it, if the application-layer protocol consists only of interactions with no or very short idle periods. • Ensure there are no long idle periods. • Resume the session after a long idle period, using 0-RTT resumption when appropriate. ... Alternatively, the client (but not the server) can use session resumption instead of sending keepalive traffic. In this case, a client that wants to send data to a server over a connection idle longer than the server's idle timeout (available from the idle timeout transport parameter) can simply reconnect. When possible, this reconnection can use 0-RTT session resumption, reducing the latency involved with restarting the connection. This of course only applies in cases in which 0-RTT data is safe, when the client is the restarting peer, and when the data to be sent is idempotent." https://www.ietf.org/id/draft-ietf-quic-applicability-07.txt
	allocate a first resource for a first non-Transmission Control Protocol (non-TCP) connection;	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.), configured to allocate a first resource (e.g. a storage and/or data structure for storing connection IDs, a connection state, send/recv buffers, etc.) for a first non-Transmission Control Protocol (non-TCP) connection (e.g., QUIC connection, etc.); See excerpt(s) above and below, for example (emphasis added, if any):

Claim	Claim Elements	Applicability
		"Google Cloud Platform (GCP), offered by Google (company), is a suite of cloud computing services that runs on the same infrastructure that Google uses internally for its end-user products, such as Google Search, Gmail, file storage, and YouTube" https://en.wikipedia.org/wiki/Google_Cloud_Platform 1. Introduction 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 transport for multiple applications. QUIC implements techniques learned from experience with TCP, SCTP and *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-07
	generate a first non-TCP packet including a first parameter field identifying first metadata for use in determining a second duration for detecting the first type of time period;	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.), configured to generate a first non-TCP packet (e.g., first QUIC negotiation packet, etc.) including a first parameter field (e.g., first idle_timeout parameter field, etc.) identifying first metadata (e.g., first idle timeout value, etc.) for use in determining a second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.) for detecting the first type of time period (e.g., idle-type timeout, etc.); See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle_timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints

Claim	Claim Elements	Applicability
		are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Value	Parameter Name	Specification
0x0000	original_connection_id	Section 18.1
0x0001	idle_timeout	Section 18.1
0x0002	stateless_reset_token	Section 18.1
0x0003	max_packet_size	Section 18.1
0x0004	initial_max_data	Section 18.1
0x0005	initial_max_stream_data_bidi_local	Section 18.1
0x0006	initial_max_stream_data_bidi_remote	Section 18.1
0x0007	initial_max_stream_data_uni	Section 18.1
0x0008	initial_max_streams_bidi	Section 18.1
0x0009	initial_max_streams_uni	Section 18.1
0x000a	ack_delay_exponent	Section 18.1
0x000b	max_ack_delay	Section 18.1
0x000c	disable_migration	Section 18.1
0x000d	preferred_address	Section 18.1
0x000e	active_connection_id_limit	Section 18.1

Table 6: Initial QUIC Transport Parameters Entries

Claim	Claim Elements	Applicability
		QUIC: A UDP-Based Multiplexed and Secure Transport https://tools.ietf.org/html/draft-ietf-quic-transport-22 "18. Transport Parameter Encoding The format of the transport parameters is the TransportParameters struct from Figure 15. This is described using the presentation language from Section 3 of [TLS13]. enum { original_connection_id(0), idle_timeout(1), stateless_reset_token(2), max_packet_size(3), initial_max_data(4), initial_max_stream_data_bidi_local(5), initial_max_stream_data_bidi_remote(6), initial_max_stream_data_uni(7), initial_max_streams_bidi(8), initial_max_streams_uni(9), ack_delay_exponent(10), max_ack_delay(11), disable_migration(12), preferred_address(13), active_connection_id_limit(14), (65535) } TransportParameterId;" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily

Claim	Claim Elements	Applicability
		modified based on the value received in the idle_timeout field of the connection negotiation packet. "idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "QUIC provides reliable, ordered delivery of the cryptographic handshake data. QUIC packet protection is used to encrypt as much of the handshake protocol as possible. The cryptographic handshake MUST provide the following properties: • authenticated key exchange, where ○ a server is always authenticated, ○ a client is optionally authenticated, ○ every connection produces distinct and unrelated keys, ○ keying material is usable for packet protection for both 0-RTT and 1-RTT packets, and ○ 1-RTT keys have forward secrecy • authenticated values for the transport parameters of the peer (see Section 7.3) • authenticated negotiation of an application protocol (TLS uses ALPN [RFC7301] for this purpose)" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	set up the first non-TCP connection, by sending, from the first node to a	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.), configured to set up the first non-TCP connection (e.g., QUIC connection, etc.), by sending, from the first node (e.g., one of a QUIC-compliant server or client, etc.) to a

Claim	Claim Elements	Applicability
	second node, the first non-TCP packet to provide the first metadata to the second node, for use by the second node in determining the second duration for detecting the first type of time period;	second node (e.g., the other one of the QUIC-compliant server or client, etc.), the first non-TCP packet (e.g., first QUIC negotiation packet, etc.) to provide the first metadata (e.g., first idle timeout value, etc.) to the second node, for use by the second node in determining the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.) for detecting the first type of time period (e.g., idle-type timeout, etc.); See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle_timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability

Value	Parameter Name	Specification
0x0000	original_connection_id	Section 18.1
0x0001	idle_timeout	Section 18.1
0x0002	stateless_reset_token	Section 18.1
0x0003	max_packet_size	Section 18.1
0x0004	initial_max_data	Section 18.1
0x0005	initial_max_stream_data_bidi_local	Section 18.1
0x0006	initial_max_stream_data_bidi_remote	Section 18.1
0x0007	initial_max_stream_data_uni	Section 18.1
0x0008	initial_max_streams_bidi	Section 18.1
0x0009	initial_max_streams_uni	Section 18.1
0x000a	ack_delay_exponent	Section 18.1
0x000b	max_ack_delay	Section 18.1
0x000c	disable_migration	Section 18.1
0x000d	preferred_address	Section 18.1
0x000e	active_connection_id_limit	Section 18.1

Table 6: Initial QUIC Transport Parameters Entries

QUIC: A UDP-Based Multiplexed and Secure Transport https://tools.ietf.org/html/draft-ietf-quic-transport-22

"QUIC provides reliable, ordered delivery of the cryptographic handshake data. QUIC packet protection is used to encrypt as much of the handshake protocol as possible. The cryptographic handshake MUST provide the following properties:

Claim	Claim Elements	Applicability
		• authenticated key exchange, where ○ a server is always authenticated, ○ a client is optionally authenticated, ○ every connection produces distinct and unrelated keys, ○ keying material is usable for packet protection for both 0-RTT and 1-RTT packets, and ○ 1-RTT keys have forward secrecy • authenticated values for the transport parameters of the peer (see Section 7.3) • authenticated negotiation of an application protocol (TLS uses ALPN [RFC7301] for this purpose)" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	in response to detecting, based on the first duration and by the first node during at least a portion of the first non-TCP connection including at least a portion of the first non-TCP connection set up, a first time period of the first type of time period, at least partially close the first non-TCP connection, by releasing, by the first node, the first	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.), configured to, in response to detecting, based on the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.) and by the first node (e.g., one of a QUIC-compliant server or client, etc.) during at least a portion of the first non-TCP connection (e.g., QUIC connection, etc.) including at least a portion of the first non-TCP connection set up, a first time period (e.g., idle timeout in connection with 0-RTT packets, etc.) of the first type of time period (e.g., idle-type timeout, etc.), at least partially close (e.g., terminate, etc.) the first non-TCP connection, by releasing, by the first node, the first resource (e.g. a storage and/or data structure for storing connection IDs, a connection state, send/recv buffers, etc.) allocated for the first non-TCP connection; See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily

Claim	Claim Elements	Applicability
	resource allocated for the first non-TCP connection;	detected based on the value received in the idle_timeout field of the connection negotiation packet. "idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10. Connection Termination An established QUIC connection can be terminated in one of three ways: • idle timeout (Section 10.2) • immediate close (Section 10.3) • stateless reset (Section 10.4) An endpoint MAY discard connection state if it does not have a validated path on which it can send packets (see Section 8.2)." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO)." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "7.3.1. Values of Transport Parameters for 0-RTT

Claim	Claim Elements	Applicability
		Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and active_connection_id_limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni

Claim	Claim Elements	Applicability
		• initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial max data and either initial max streams bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	in response to detecting, based on the second duration and by the first node after the first duration is changed to the second duration, a second	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.), configured to, in response to detecting, based on the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.) and by the first node (e.g., one of a QUIC-compliant server or client, etc.) after the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.) is changed to the second duration, a second time period (e.g., idle timeout in connection with 1-RTT

Claim	Claim Elements	Applicability
	time period of the first type of time period, at least partially close the first non-TCP connection, by releasing, by the first node, the first resource allocated for the first non-TCP connection; and	packets, etc.) of the first type of time period (e.g., idle-type timeout, etc.), at least partially close (e.g., terminate, etc.) the first non-TCP connection (e.g., QUIC connection, etc.), by releasing, by the first node, the first resource (e.g. a storage and/or data structure for storing connection IDs, a connection state, send/recv buffers, etc.) allocated for the first non-TCP connection; and See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily detected based on the value received in the idle_timeout field of the connection negotiation packet. "idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10. Connection Termination An established QUIC connection can be terminated in one of three ways: • idle timeout (Section 10.2) • immediate close (Section 10.3) • stateless reset (Section 10.4) An endpoint MAY discard connection state if it does not have a validated path on which it can send packets (see Section 8.2)." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout

Claim	Claim Elements	Applicability
		If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake.

Claim	Claim Elements	Applicability
		The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original connection id, preferred address, stateless reset token, ack delay exponent and active connection id limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial_max_data and either initial_max_streams_bidi and

Claim	Claim Elements	Applicability
		initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	in response to detecting, based on a third duration, a third time period, at least partially close the first non-TCP connection, by releasing, by the first node, the first resource allocated for the first non-TCP connection, where the third duration is determined based on a first algorithm that is different from a second algorithm on which a determination	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.), configured to, in response to detecting, based on a third duration (e.g., duration associated with probe timeout, etc.), a third time period (e.g., probe timeout, etc.), at least partially close (e.g., terminate, etc.) the first non-TCP connection (e.g., QUIC connection, etc.), by releasing, by the first node (e.g., one of a QUIC-compliant server or client, etc.), the first resource (e.g. a storage and/or data structure for storing connection IDs, a connection state, send/recv buffers, etc.) allocated for the first non-TCP connection, where the third duration is determined based on a first algorithm (e.g., used to determine the probe timeout, etc.) that is different from a second algorithm (e.g., used to negotiate an idle timeout period between nodes, etc.) on which a determination of the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.) is based. See excerpt(s) above and below, for example (emphasis added, if any):

Claim	Claim Elements	Applicability
	of the second duration is based.	"10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to

Claim	Claim Elements	Applicability
		1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement.

Claim	Claim Elements	Applicability
		The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3
2	The apparatus of claim 1, wherein the apparatus is configured such that the first duration is not negotiated between the first and second nodes during the first non-TCP connection set up, and the second duration is negotiated between the first and second nodes during the first non-TCP connection set up for the first non-TCP connection.	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 1 and is configured such that the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.) is not negotiated between the first and second nodes during the first non-TCP connection (e.g., QUIC connection, etc.) set up, and the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.) is negotiated between the first and second nodes during the first non-TCP connection set up for the first non-TCP connection. See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints

Claim	Claim Elements	Applicability
		are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "7.3.1. Values of Transport Parameters for 0-RTT

Claim	Claim Elements	Applicability
		Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and active_connection_id_limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni

Claim	Claim Elements	Applicability
		• initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial max data and either initial max streams bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
3	The apparatus of claim 1, wherein the apparatus is configured such that the first time period is only capable of being detected	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 1 and is configured such that the first time period (e.g., idle timeout in connection with 0-RTT packets, etc.) is only capable of being detected during the first non-TCP connection (e.g., QUIC connection, etc.) set up. See excerpt(s) above and below, for example (emphasis added, if any):

Claim	Claim Elements	Applicability
	during the first non-TCP connection set up.	**Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original connection id, preferred address, stateless reset token, ack delay exponent and active connection id limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value.

Claim	Claim Elements	Applicability
		A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial max data and either initial max streams bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of

Claim	Claim Elements	Applicability
		transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
6	The apparatus of claim 1, wherein the apparatus is configured such that the third time period is detected based on the third duration, by being detected when no acknowledgement packet is received by the first node in the first non-TCP connection and processed as an acknowledgement, during the third duration, for a sent packet sent by the first node in the first non-TCP connection.	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 1 and is configured such that the third time period (e.g., probe timeout, etc.) is detected based on the third duration (e.g., duration associated with probe timeout, etc.), by being detected when no acknowledgement packet is received by the first node (e.g., one of a QUIC-compliant server or client, etc.) in the first non-TCP connection (e.g., QUIC connection, etc.) and processed as an acknowledgement, during the third duration, for a sent packet sent by the first node in the first non-TCP connection. See excerpt(s) above and below, for example (emphasis added, if any): "18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout

Claim	Claim Elements	Applicability
		A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion.

Claim	Claim Elements	Applicability
		A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
10	The apparatus of claim 1, wherein the apparatus is	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 1 and is configured such that the first non-TCP packet (e.g.,

Claim	Claim Elements	Applicability
	configured such that the first non-TCP packet also includes data, separate from the first metadata, for use in determining the third time period.	first QUIC negotiation packet, etc.) also includes data (e.g. metadata exchanged for setting max_ack_delay, etc.), separate from the first metadata (e.g., first idle timeout value, etc.), for use in determining the third time period (e.g., probe timeout, etc.). See excerpt(s) above and below, for example (emphasis added, if any): "18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows:

Claim	Claim Elements	Applicability
		PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO).

Claim	Claim Elements	Applicability
		Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
12	The apparatus of claim 1, wherein the apparatus is configured such that the first duration is not negotiated between the first and second nodes during the first non-TCP connection set up, the second duration is negotiated during the first non-TCP connection set up between the first and second nodes for the first	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 1 and is configured such that the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.) is not negotiated between the first and second nodes during the first non-TCP connection (e.g., QUIC connection, etc.) set up, the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.) is negotiated during the first non-TCP connection set up between the first and second nodes for the first non-TCP connection, and the third duration (e.g., duration associated with probe timeout, etc.) of the third time period (e.g., probe timeout, etc.) is negotiated during the first non-TCP connection set up between the first and second nodes for the first non-TCP connection (e.g. the third time period is calculated based on the max_ack_delay parameter, etc.). See excerpt(s) above and below, for example (emphasis added, if any):

Claim	Claim Elements	Applicability
	non-TCP connection, and the third duration of the third time period is negotiated during the first non-TCP connection set up between the first and second nodes for the first non-TCP connection.	**Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and

Claim	Claim Elements	Applicability
		F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3

Claim	Claim Elements	Applicability
		"10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT

Claim	Claim Elements	Applicability
		packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and active_connection_id limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni

Claim	Claim Elements	Applicability
		Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial_max_data and either initial_max_streams_bidi and initial_max_stream_data_bidi_remote, or initial_max_streams_uni and initial_max_stream_data_uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
15	The apparatus of claim 1, wherein the apparatus is configured such that the third time period is a second type of time period.	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 1 and is configured such that the third time period (e.g., probe timeout, etc.) is a second type of time period (e.g., probe-type timeout, etc.). See excerpt(s) above and below, for example (emphasis added, if any): "18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending

Claim	Claim Elements	Applicability
		acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement.

Claim	Claim Elements	Applicability
		The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded

Claim	Claim Elements	Applicability
		if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
20	The apparatus of claim 1, wherein the apparatus is configured such that the third duration is determined using a second parameter field identifying second metadata, where the second parameter field is communicated in the first non-TCP connection.	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 1 and is configured such that the third duration (e.g., duration associated with probe timeout, etc.) is determined using a second parameter field (e.g., second idle_timeout parameter field, etc.) identifying second metadata (e.g., second idle timeout value, etc.), where the second parameter field is communicated in the first non-TCP connection (e.g., QUIC connection, etc.). See excerpt(s) above and below, for example (emphasis added, if any): "18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout

Claim	Claim Elements	Applicability
		A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion.

Claim	Claim Elements	Applicability
		A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily

Claim	Claim Elements	Applicability
		modified based on the value received in the idle_timeout field of the connection negotiation packet. "idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
21	The apparatus of claim 20, wherein the apparatus is configured such that the second duration is determined using the first parameter field identifying the first metadata, and the second parameter field identifying the second metadata.	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 20 and is configured such that the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.) is determined using the first parameter field (e.g., first idle_timeout parameter field, etc.) identifying the first metadata (e.g., first idle timeout value, etc.), and the second parameter field (e.g., second idle_timeout parameter field, etc.) identifying the second metadata (e.g., second idle timeout value, etc.). See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
		"18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3.

Claim	Claim Elements	Applicability
		The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity.

Claim	Claim Elements	Applicability
		Note: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily modified based on the value received in the idle_timeout field of the connection negotiation packet. "idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
27	The apparatus of claim 1, wherein the apparatus is configured such that the first type of time period includes an idle time period.	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 1 and is configured such that the first type of time period (e.g., idle-type timeout, etc.) includes an idle time period. See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle_timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability

Value	Parameter Name	Specification
0x0000	original_connection_id	Section 18.1
0x0001	idle_timeout	Section 18.1
0x0002	stateless_reset_token	Section 18.1
0x0003	max_packet_size	Section 18.1
0x0004	initial_max_data	Section 18.1
0x0005	initial_max_stream_data_bidi_local	Section 18.1
0x0006	initial_max_stream_data_bidi_remote	Section 18.1
0x0007	initial_max_stream_data_uni	Section 18.1
0x0008	initial_max_streams_bidi	Section 18.1
0x0009	initial_max_streams_uni	Section 18.1
0x000a	ack_delay_exponent	Section 18.1
0x000b	max_ack_delay	Section 18.1
0x000c	disable_migration	Section 18.1
0x000d	preferred_address	Section 18.1
0x000e	active_connection_id_limit	Section 18.1

Table 6: Initial QUIC Transport Parameters Entries

QUIC: A UDP-Based Multiplexed and Secure Transport https://tools.ietf.org/html/draft-ietf-quic-transport-22

"18. Transport Parameter Encoding

Claim	Claim Elements	Applicability
		The format of the transport parameters is the TransportParameters struct from Figure 15. This is described using the presentation language from Section 3 of [TLS13]. enum { original_connection_id(0), idle_timeout(1), stateless_reset_token(2), max_packet_size(3), initial_max_data(4), initial_max_stream_data_bidi_local(5), initial_max_stream_data_bidi_remote(6), initial_max_stream_data_uni(7), initial_max_streams_bidi(8), initial_max_streams_uni(9), ack_delay_exponent(10), max_ack_delay(11), disable_migration(12), preferred_address(13), active_connection_id_limit(14), (65535) } TransportParameterId;" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily modified based on the value received in the idle_timeout field of the connection negotiation packet.

Claim	Claim Elements	Applicability
		"idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
28	The apparatus of claim 27, wherein the apparatus is configured such that the idle time period is one: during which, no packet is communicated in the non-TCP connection to keep the non-TCP connection active.	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 27 and is configured such that the idle time period is one: during which, no packet is communicated in the non-TCP connection (e.g., QUIC connection, etc.) to keep the non-TCP connection active. See excerpt(s) above and below, for example (emphasis added, if any): "A connection will time out if no packets are sent or received for a period longer than the time negotiated using the max_idle_timeout transport parameter (see Section 10). However, state in middleboxes might time out earlier than that. Though REQ-5 in [RFC4787] recommends a 2 minute timeout interval, experience shows that sending packets every 15 to 30 seconds is necessary to prevent the majority of middleboxes from losing state for UDP flows." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/id/draft-ietf-quic-transport-27.txt
35	The apparatus of claim 28, wherein the apparatus is configured such that the idle time period is one: during which, no packet is sent or received by the apparatus in the non-TCP connection to keep the	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 28 and is configured such that the idle time period is one: during which, no packet is sent or received by the apparatus in the non-TCP connection (e.g., QUIC connection, etc.) to keep the non-TCP connection active. See excerpt(s) above and below, for example (emphasis added, if any): "A connection will time out if no packets are sent or received for a period longer than the time negotiated using the max_idle_timeout transport parameter (see Section 10). However,

Claim	Claim Elements	Applicability
	non-TCP connection active.	state in middleboxes might time out earlier than that. Though REQ-5 in [RFC4787] recommends a 2 minute timeout interval, experience shows that sending packets every 15 to 30 seconds is necessary to prevent the majority of middleboxes from losing state for UDP flows." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/id/draft-ietf-quic-transport-27.txt
39	The apparatus of claim 1, wherein the apparatus is configured such that, during the at least portion of the first non-TCP connection including the at least portion of the first non-TCP connection set up, the detection of the first time period of the first type of time period is based on the first duration, by being only based on whether another non-TCP packet is not received for the first non-TCP connection set up for the first duration.	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 1 and is configured such that, during the at least portion of the first non-TCP connection (e.g., QUIC connection, etc.) including the at least portion of the first non-TCP connection set up, the detection of the first time period (e.g., idle timeout in connection with 0-RTT packets, etc.) of the first type of time period (e.g., idle-type timeout, etc.) is based on the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.), by being only based on whether another non-TCP packet is not received for the first non-TCP connection set up for the first duration. See excerpt(s) above and below, for example (emphasis added, if any): "A connection will time out if no packets are sent or received for a period longer than the time negotiated using the max_idle_timeout transport parameter (see Section 10). However, state in middleboxes might time out earlier than that. Though REQ-5 in [RFC4787] recommends a 2 minute timeout interval, experience shows that sending packets every 15 to 30 seconds is necessary to prevent the majority of middleboxes from losing state for UDP flows." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/id/draft-ietf-quic-transport-27.txt **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters

Claim	Claim Elements	Applicability
		During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and active_connection_id_limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data.

Claim	Claim Elements	Applicability
		If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial max data and either initial max streams bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL VIOLATION.”

Claim	Claim Elements	Applicability
		QUIC: A UDP-Based Multiplexed and Secure Transport https://tools.ietf.org/html/draft-ietf-quic-transport-22
48	The apparatus of claim 1, wherein the apparatus is configured such that the first type of time period is started in response to at least one of a received packet or a sent packet.	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 1 and is configured such that the first type of time period (e.g., idle-type timeout, etc.) is started in response to at least one of a received packet or a sent packet. See excerpt(s) above and below, for example (emphasis added, if any): "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried."

Claim	Claim Elements	Applicability
		QUIC: A UDP-Based Multiplexed and Secure Transport https://tools.ietf.org/html/draft-ietf-quic-transport-22 "QUIC provides reliable, ordered delivery of the cryptographic handshake data. QUIC packet protection is used to encrypt as much of the handshake protocol as possible. The cryptographic handshake MUST provide the following properties: • authenticated key exchange, where ○ a server is always authenticated, ○ a client is optionally authenticated, ○ every connection produces distinct and unrelated keys, ○ keying material is usable for packet protection for both 0-RTT and 1-RTT packets, and ○ 1-RTT keys have forward secrecy • authenticated values for the transport parameters of the peer (see Section 7.3) • authenticated negotiation of an application protocol (TLS uses ALPN [RFC7301] for this purpose)" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
51	The apparatus of claim 48, wherein the apparatus is configured such that, after the first duration is changed to the second duration, the detection of the second time period of the first type of time period is based on the	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 48 and is configured such that, after the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.) is changed to the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.), the detection of the second time period (e.g., idle timeout in connection with 1-RTT packets, etc.) of the first type of time period (e.g., idle-type timeout, etc.) is based on the second duration, by being only based on whether, for the second duration after the second time period of the first type of time period is started, no non-empty packet is received (e.g.,

Claim	Claim Elements	Applicability
	second duration, by being only based on whether, for the second duration after the second time period of the first type of time period is started, no non-empty packet is received and processed to keep the first non-TCP connection at least partially active.	connection remains idle, etc.) and processed to keep the first non-TCP connection (e.g., QUIC connection, etc.) at least partially active. See excerpt(s) above and below, for example (emphasis added, if any): "A connection will time out if no packets are sent or received for a period longer than the time negotiated using the max_idle_timeout transport parameter (see Section 10). However, state in middleboxes might time out earlier than that. Though REQ-5 in [RFC4787] recommends a 2 minute timeout interval, experience shows that sending packets every 15 to 30 seconds is necessary to prevent the majority of middleboxes from losing state for UDP flows." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/id/draft-ietf-quic-transport-27.txt **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily detected based on the value received in the idle_timeout field of the connection negotiation packet. "idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling.

Claim	Claim Elements	Applicability
		7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and active_connection_id_limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local

Claim	Claim Elements	Applicability
		• initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial max data and either initial max streams bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10. Connection Termination An established QUIC connection can be terminated in one of three ways: • idle timeout (Section 10.2)

Claim	Claim Elements	Applicability
		• immediate close (Section 10.3) • stateless reset (Section 10.4) An endpoint MAY discard connection state if it does not have a validated path on which it can send packets (see Section 8.2)." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
		"QUIC provides reliable, ordered delivery of the cryptographic handshake data. QUIC packet protection is used to encrypt as much of the handshake protocol as possible. The cryptographic handshake MUST provide the following properties: • authenticated key exchange, where ○ a server is always authenticated, ○ a client is optionally authenticated, ○ every connection produces distinct and unrelated keys, ○ keying material is usable for packet protection for both 0-RTT and 1-RTT packets, and ○ 1-RTT keys have forward secrecy • authenticated values for the transport parameters of the peer (see Section 7.3) • authenticated negotiation of an application protocol (TLS uses ALPN [RFC7301] for this purpose)" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
52	The apparatus of claim 51, wherein the apparatus is configured such that, during the second time period of the first type of time period, an empty packet is sent without causing a timer utilized to detect the second time period to be reset.	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 51 and is configured such that, during the second time period (e.g., idle timeout in connection with 1-RTT packets, etc.) of the first type of time period (e.g., idle-type timeout, etc.), an empty packet is sent (i.e. a packet carries no application data, etc.) without causing a timer utilized to detect the second time period to be reset. See excerpt(s) above and below, for example (emphasis added, if any): **Note**: Based upon information and belief, keep-alive signals are not required as a result of using idle timers and reducing the number of keep-alive signals to zero and, without a negotiated idle timeout, neither endpoint of a connection knows when the other side might timeout, so keep-alives (e.g., PINGs) have to be sent at unnecessarily shorter intervals to

Claim	Claim Elements	Applicability
		insure the connection to stay active. Further, with a negotiated idle time period, only one side sends one or more keep-alives, if a keep-alive is used. The QUIC IETF transport spec recognizes the need to avoid unnecessary keep-alives (e.g., PINGs). See section 10.1.2 of the QUIC transport draft at https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-34#section-10.1.2 and section 5.1 of the IETF HTTP/3 draft at https://tools.ietf.org/id/draft-ietf-quic-http-34.html#name-idle-connections. "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
		"19.2. PING Frame Endpoints can use PING frames (type=0x01) to verify that their peers are still alive or to check reachability to the peer. The PING frame contains no additional fields. The receiver of a PING frame simply needs to acknowledge the packet containing this frame. The PING frame can be used to keep a connection alive when an application or application protocol wishes to prevent the connection from timing out. An application protocol SHOULD provide guidance about the conditions under which generating a PING is recommended. This guidance SHOULD indicate whether it is the client or the server that is expected to send the PING. Having both endpoints send PING frames without coordination can produce an excessive number of packets and poor performance. A connection will time out if no packets are sent or received for a period longer than the time negotiated using the max_idle_timeout transport parameter (see Section 10). However, state in middleboxes might time out earlier than that. Though REQ-5 in [RFC4787] recommends a 2 minute timeout interval, experience shows that sending packets every 15 to 30 seconds is necessary to prevent the majority of middleboxes from losing state for UDP flows." https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-27#section-19.2
55	The apparatus of claim 1, wherein the apparatus is configured such that, after the first duration is changed to the second duration, the detection of the second time period of the first type of time period is based on the	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 1 and is configured such that, after the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.) is changed to the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.), the detection of the second time period (e.g., idle timeout in connection with 1-RTT packets, etc.) of the first type of time period (e.g., idle-type timeout, etc.) is based on the second duration, by being only based on whether, for the second duration, no non-empty packet is received (e.g., connection remains idle, etc.) and processed to keep the first non-TCP connection (e.g., QUIC connection, etc.) at least partially active.

Claim	Claim Elements	Applicability
	second duration, by being only based on whether, for the second duration, no non-empty packet is received and processed to keep the first non-TCP connection at least partially active.	See excerpt(s) above and below, for example (emphasis added, if any): "A connection will time out if no packets are sent or received for a period longer than the time negotiated using the max_idle_timeout transport parameter (see Section 10). However, state in middleboxes might time out earlier than that. Though REQ-5 in [RFC4787] recommends a 2 minute timeout interval, experience shows that sending packets every 15 to 30 seconds is necessary to prevent the majority of middleboxes from losing state for UDP flows." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/id/draft-ietf-quic-transport-27.txt **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily detected based on the value received in the idle_timeout field of the connection negotiation packet. "idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling. 7.3.1. Values of Transport Parameters for 0-RTT

Claim	Claim Elements	Applicability
		Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and active connection id limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote

Claim	Claim Elements	Applicability
		• initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial_max_data and either initial_max_streams_bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10. Connection Termination An established QUIC connection can be terminated in one of three ways: • idle timeout (Section 10.2) • immediate close (Section 10.3)

Claim	Claim Elements	Applicability
		• stateless reset (Section 10.4) An endpoint MAY discard connection state if it does not have a validated path on which it can send packets (see Section 8.2)." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
		"QUIC provides reliable, ordered delivery of the cryptographic handshake data. QUIC packet protection is used to encrypt as much of the handshake protocol as possible. The cryptographic handshake MUST provide the following properties: • authenticated key exchange, where ○ a server is always authenticated, ○ a client is optionally authenticated, ○ every connection produces distinct and unrelated keys, ○ keying material is usable for packet protection for both 0-RTT and 1-RTT packets, and ○ 1-RTT keys have forward secrecy • authenticated values for the transport parameters of the peer (see Section 7.3) • authenticated negotiation of an application protocol (TLS uses ALPN [RFC7301] for this purpose)" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
56	The apparatus of claim 55, wherein the apparatus is configured such that, during the second time period of the first type of time period, an empty packet is sent without causing a timer utilized to detect the second time period to be reset.	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 55 and is configured such that, during the second time period (e.g., idle timeout in connection with 1-RTT packets, etc.) of the first type of time period (e.g., idle-type timeout, etc.), an empty packet is sent (i.e. a packet carries no application data, etc.) without causing a timer utilized to detect the second time period to be reset. See excerpt(s) above and below, for example (emphasis added, if any): **Note**: Based upon information and belief, keep-alive signals are not required as a result of using idle timers and reducing the number of keep-alive signals to zero and, without a negotiated idle timeout, neither endpoint of a connection knows when the other side might timeout, so keep-alives (e.g., PINGs) have to be sent at unnecessarily shorter intervals to

Claim	Claim Elements	Applicability
		insure the connection to stay active. Further, with a negotiated idle time period, only one side sends one or more keep-alives, if a keep-alive is used. The QUIC IETF transport spec recognizes the need to avoid unnecessary keep-alives (e.g., PINGs). See section 10.1.2 of the QUIC transport draft at https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-34#section-10.1.2 and section 5.1 of the IETF HTTP/3 draft at https://tools.ietf.org/id/draft-ietf-quic-http-34.html#name-idle-connections. "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
		"19.2. PING Frame Endpoints can use PING frames (type=0x01) to verify that their peers are still alive or to check reachability to the peer. The PING frame contains no additional fields. The receiver of a PING frame simply needs to acknowledge the packet containing this frame. The PING frame can be used to keep a connection alive when an application or application protocol wishes to prevent the connection from timing out. An application protocol SHOULD provide guidance about the conditions under which generating a PING is recommended. This guidance SHOULD indicate whether it is the client or the server that is expected to send the PING. Having both endpoints send PING frames without coordination can produce an excessive number of packets and poor performance. A connection will time out if no packets are sent or received for a period longer than the time negotiated using the max_idle_timeout transport parameter (see Section 10). However, state in middleboxes might time out earlier than that. Though REQ-5 in [RFC4787] recommends a 2 minute timeout interval, experience shows that sending packets every 15 to 30 seconds is necessary to prevent the majority of middleboxes from losing state for UDP flows." https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-27#section-19.2
63	The apparatus of claim 1, wherein the apparatus is configured such that, during the at least portion of the first non-TCP connection including the at least portion of the first non-TCP connection set up, the detection of the	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 1 and is configured such that, during the at least portion of the first non-TCP connection (e.g., QUIC connection, etc.) including the at least portion of the first non-TCP connection set up, the detection of the first time period (e.g., idle timeout in connection with 0-RTT packets, etc.) of the first type of time period (e.g., idle-type timeout, etc.) is based on the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.), by being based on whether, for the first duration, no packet is received (e.g., connection remains idle, etc.) and processed to keep the first non-TCP connection at least partially active.

Claim	Claim Elements	Applicability
	first time period of the first type of time period is based on the first duration, by being based on whether, for the first duration, no packet is received and processed to keep the first non-TCP connection at least partially active.	See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle timeout parameter that is detected by a recipient of such packet. "A connection will time out if no packets are sent or received for a period longer than the time negotiated using the max_idle_timeout transport parameter (see Section 10). However, state in middleboxes might time out earlier than that. Though REQ-5 in [RFC4787] recommends a 2 minute timeout interval, experience shows that sending packets every 15 to 30 seconds is necessary to prevent the majority of middleboxes from losing state for UDP flows." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/id/draft-ietf-quic-transport-27.txt "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling. 7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake.

Claim	Claim Elements	Applicability
		The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original connection id, preferred address, stateless reset token, ack delay exponent and active connection id limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial_max_data and either initial_max_streams_bidi and

Claim	Claim Elements	Applicability
		initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity.

Claim	Claim Elements	Applicability
		The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
64	The apparatus of claim 1, wherein the apparatus is configured such that the first duration is not negotiated during the first non-TCP connection set up between the first and second nodes for the first non-TCP connection, and the second duration is negotiated between the first and second nodes for the first non-TCP connection.	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 1 and is configured such that the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.) is not negotiated during the first non-TCP connection (e.g., QUIC connection, etc.) set up between the first and second nodes for the first non-TCP connection, and the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.) is negotiated between the first and second nodes for the first non-TCP connection. See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling. 7.3.1. Values of Transport Parameters for 0-RTT

Claim	Claim Elements	Applicability
		Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and active connection id limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote

Claim	Claim Elements	Applicability
		• initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial_max_data and either initial_max_streams_bidi and initial_max_stream_data_bidi_remote, or initial_max_streams_uni and initial_max_stream_data_uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO).

Claim	Claim Elements	Applicability
		Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "QUIC provides reliable, ordered delivery of the cryptographic handshake data. QUIC packet protection is used to encrypt as much of the handshake protocol as possible. The cryptographic handshake MUST provide the following properties: • authenticated key exchange, where ○ a server is always authenticated, ○ a client is optionally authenticated, ○ every connection produces distinct and unrelated keys, ○ keying material is usable for packet protection for both 0-RTT and 1-RTT packets, and ○ 1-RTT keys have forward secrecy • authenticated values for the transport parameters of the peer (see Section 7.3)

Claim	Claim Elements	Applicability
		• authenticated negotiation of an application protocol (TLS uses ALPN [RFC7301] for this purpose)" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
65	The apparatus of claim 1, wherein the apparatus is configured to, in response to detecting, by the first node, the third time period that is a second type of time period including an acknowledgment timeout period during which no acknowledgement packet is received in the first non-TCP connection and processed as an acknowledgement for a sent packet sent in the first non-TCP connection:	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 1 and is configured wherein the apparatus is configured to, in response to detecting, by the first node (e.g., one of a QUIC-compliant server or client, etc.), the third time period (e.g., probe timeout, etc.) that is a second type of time period (e.g., probe-type timeout, etc.) including an acknowledgment timeout period during which no acknowledgement packet is received in the first non-TCP connection (e.g., QUIC connection, etc.) and processed as an acknowledgement for a sent packet sent in the first non-TCP connection: See excerpt(s) above and below, for example (emphasis added, if any): "18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout

Claim	Claim Elements	Applicability
		A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion.

Claim	Claim Elements	Applicability
		A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	unilaterally at least partially close the first	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 1 and is configured to unilaterally at least partially close (e.g.,

Claim	Claim Elements	Applicability
	non-TCP connection, by releasing, by the first node, the first resource allocated for the first non-TCP connection,	terminate, etc.) the first non-TCP connection (e.g., QUIC connection, etc.), by releasing, by the first node (e.g., one of a QUIC-compliant server or client, etc.), the first resource (e.g. a storage and/or data structure for storing connection IDs, a connection state, send/recv buffers, etc.) allocated for the first non-TCP connection, See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily detected based on the value received in the idle_timeout field of the connection negotiation packet. "idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10. Connection Termination An established QUIC connection can be terminated in one of three ways: • idle timeout (Section 10.2) • immediate close (Section 10.3) • stateless reset (Section 10.4) An endpoint MAY discard connection state if it does not have a validated path on which it can send packets (see Section 8.2)." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout

Claim	Claim Elements	Applicability
		If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO

Claim	Claim Elements	Applicability
		When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3
	wherein: the first non-TCP connection set up further includes, in addition to (a-	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 1 and is configured wherein: the first non-TCP connection (e.g., QUIC connection, etc.) set up further includes, in addition to (a-1) sending the first

Claim	Claim Elements	Applicability
	1) sending the first non-TCP packet that is also for use in determining the third duration of the third time period:	non-TCP packet (e.g., first QUIC negotiation packet, etc.) that is also for use in determining the third duration (e.g., duration associated with probe timeout, etc.) of the third time period (e.g., probe timeout, etc.): See excerpt(s) above and below, for example (emphasis added, if any): "18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows:

Claim	Claim Elements	Applicability
		PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO).

Claim	Claim Elements	Applicability
		Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	(a-2) after sending the first non-TCP packet and without any other communication between the first node and the second node after (a-1) and before (a-2), receiving, at the first node from the second node, a second non-TCP packet, and (a-3) after receiving the second non-TCP packet and without any other communication	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 1 and is configured wherein (a-2) after sending the first non-TCP packet (e.g., first QUIC negotiation packet, etc.) and without any other communication between the first node (e.g., one of a QUIC-compliant server or client, etc.) and the second node (e.g., the other one of the QUIC-compliant server or client, etc.) after (a-1) and before (a-2), receiving, at the first node from the second node, a second non-TCP packet (e.g., second QUIC negotiation packet, etc.), and (a-3) after receiving the second non-TCP packet and without any other communication between the first node and the second node after (a-2) and before (a-3), sending, from the first node to the second node, a third non-TCP packet, and further wherein: See excerpt(s) above and below, for example (emphasis added, if any):

Claim	Claim Elements	Applicability
	between the first node and the second node after (a-2) and before (a-3), sending, from the first node to the second node, a third non-TCP packet, and further wherein:	**Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle_timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
		+--------+--------------------------------------+---------------+ | Value | Parameter Name | Specification | +--------+--------------------------------------+---------------+ | 0x0000 | original_connection_id | Section 18.1 | | 0x0001 | idle_timeout | Section 18.1 | | 0x0002 | stateless_reset_token | Section 18.1 | | 0x0003 | max_packet_size | Section 18.1 | | 0x0004 | initial_max_data | Section 18.1 | | 0x0005 | initial_max_stream_data_bidi_local | Section 18.1 | | 0x0006 | initial_max_stream_data_bidi_remote | Section 18.1 | | 0x0007 | initial_max_stream_data_uni | Section 18.1 | | 0x0008 | initial_max_streams_bidi | Section 18.1 | | 0x0009 | initial_max_streams_uni | Section 18.1 | | 0x000a | ack_delay_exponent | Section 18.1 | | 0x000b | max_ack_delay | Section 18.1 | | 0x000c | disable_migration | Section 18.1 | | 0x000d | preferred_address | Section 18.1 | | 0x000e | active_connection_id_limit | Section 18.1 | +--------+--------------------------------------+---------------+ Table 6: Initial QUIC Transport Parameters Entries *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	the first duration, the second duration, and the	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 1 and is configured wherein the first duration (e.g., duration

Claim	Claim Elements	Applicability
	third duration are capable of being of different durations;	associated with idle timeout in connection with 0-RTT packets, etc.), the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.), and the third duration (e.g., duration associated with probe timeout, etc.) are capable of being of different durations; See excerpt(s) above and below, for example (emphasis added, if any): "The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity.

Claim	Claim Elements	Applicability
		The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and active_connection_id_limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-

Claim	Claim Elements	Applicability
		protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial max data and either initial max streams bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if

Claim	Claim Elements	Applicability
		those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "QUIC provides reliable, ordered delivery of the cryptographic handshake data. QUIC packet protection is used to encrypt as much of the handshake protocol as possible. The cryptographic handshake MUST provide the following properties: • authenticated key exchange, where ○ a server is always authenticated, ○ a client is optionally authenticated, ○ every connection produces distinct and unrelated keys, ○ keying material is usable for packet protection for both 0-RTT and 1-RTT packets, and ○ 1-RTT keys have forward secrecy • authenticated values for the transport parameters of the peer (see Section 7.3) • authenticated negotiation of an application protocol (TLS uses ALPN [RFC7301] for this purpose)" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	the second time period and the third time period are capable of being detected during the first non-TCP connection set up and	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 1 and is configured wherein the second time period (e.g., idle timeout in connection with 1-RTT packets, etc.) and the third time period (e.g., probe timeout, etc.) are capable of being detected during the first non-TCP connection (e.g., QUIC connection, etc.) set up and thereafter during the first non-TCP connection; and

Claim	Claim Elements	Applicability
	thereafter during the first non-TCP connection; and	See excerpt(s) above and below, for example (emphasis added, if any): "18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay

Claim	Claim Elements	Applicability
		kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an

Claim	Claim Elements	Applicability
		ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	the first time period and the second time period are started in response to at least one of a received packet or a sent packet; and	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 1 and is configured wherein the first time period (e.g., idle timeout in connection with 0-RTT packets, etc.) and the second time period (e.g., idle timeout in connection with 1-RTT packets, etc.) are started in response to at least one of a received packet or a sent packet; and See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling."

Claim	Claim Elements	Applicability
		QUIC: A UDP-Based Multiplexed and Secure Transport https://tools.ietf.org/html/draft-ietf-quic-transport-22 "18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay

Claim	Claim Elements	Applicability
		kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an

Claim	Claim Elements	Applicability
		ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	during the at least portion of the first non-TCP connection including the at least portion of the first non-TCP connection set up, the detection of the first time period of the first type of time period is based on the first duration, by being based on whether, during the first duration, no packet is received and processed to keep the first non-TCP connection at least partially active; and	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 1 and is configured wherein, during the at least portion of the first non-TCP connection (e.g., QUIC connection, etc.) including the at least portion of the first non-TCP connection set up, the detection of the first time period (e.g., idle timeout in connection with 0-RTT packets, etc.) of the first type of time period (e.g., idle-type timeout, etc.) is based on the first duration, by being based on whether, during the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.), no packet is received (e.g., connection remains idle, etc.) and processed to keep the first non-TCP connection at least partially active; and See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle timeout parameter that is detected by a recipient of such packet.

Claim	Claim Elements	Applicability
		"A connection will time out if no packets are sent or received for a period longer than the time negotiated using the max_idle_timeout transport parameter (see Section 10). However, state in middleboxes might time out earlier than that. Though REQ-5 in [RFC4787] recommends a 2 minute timeout interval, experience shows that sending packets every 15 to 30 seconds is necessary to prevent the majority of middleboxes from losing state for UDP flows." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/id/draft-ietf-quic-transport-27.txt "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling. 7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and

Claim	Claim Elements	Applicability
		active connection id limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial max data and either initial max streams bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported.

Claim	Claim Elements	Applicability
		When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried."

Claim	Claim Elements	Applicability
		QUIC: A UDP-Based Multiplexed and Secure Transport https://tools.ietf.org/html/draft-ietf-quic-transport-22
	after the first duration is changed to the second duration, the detection of the second time period of the first type of time period is based on the second duration, by being based on whether, during the second duration, no packet is received and processed to keep the first non-TCP connection at least partially active;	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 1 and is configured wherein, after the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.) is changed to the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.), the detection of the second time period (e.g., idle timeout in connection with 1-RTT packets, etc.) of the first type of time period (e.g., idle-type timeout, etc.) is based on the second duration, by being based on whether, during the second duration, no packet is received (e.g., connection remains idle, etc.) and processed to keep the first non-TCP connection (e.g., QUIC connection, etc.) at least partially active; See excerpt(s) above and below, for example (emphasis added, if any): "A connection will time out if no packets are sent or received for a period longer than the time negotiated using the max_idle_timeout transport parameter (see Section 10). However, state in middleboxes might time out earlier than that. Though REQ-5 in [RFC4787] recommends a 2 minute timeout interval, experience shows that sending packets every 15 to 30 seconds is necessary to prevent the majority of middleboxes from losing state for UDP flows." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/id/draft-ietf-quic-transport-27.txt **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily detected based on the value received in the idle_timeout field of the connection negotiation packet.

Claim	Claim Elements	Applicability
		"idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling. 7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original connection id, preferred address, stateless reset token, ack delay exponent and active connection id limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value.

Claim	Claim Elements	Applicability
		A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial max data and either initial max streams bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of

Claim	Claim Elements	Applicability
		transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10. Connection Termination An established QUIC connection can be terminated in one of three ways: • idle timeout (Section 10.2) • immediate close (Section 10.3) • stateless reset (Section 10.4) An endpoint MAY discard connection state if it does not have a validated path on which it can send packets (see Section 8.2)." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity.

Claim	Claim Elements	Applicability
		The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "QUIC provides reliable, ordered delivery of the cryptographic handshake data. QUIC packet protection is used to encrypt as much of the handshake protocol as possible. The cryptographic handshake MUST provide the following properties: • authenticated key exchange, where ○ a server is always authenticated, ○ a client is optionally authenticated, ○ every connection produces distinct and unrelated keys, ○ keying material is usable for packet protection for both 0-RTT and 1-RTT packets, and ○ 1-RTT keys have forward secrecy • authenticated values for the transport parameters of the peer (see Section 7.3) • authenticated negotiation of an application protocol (TLS uses ALPN [RFC7301] for this purpose)" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
	where only one of the first time period, the second time period, or the third time period is detected for the first non-TCP connection.	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 1 and is configured where only one of the first time period (e.g., idle timeout in connection with 0-RTT packets, etc.), the second time period (e.g., idle timeout in connection with 1-RTT packets, etc.), or the third time period (e.g., probe timeout, etc.) is detected for the first non-TCP connection (e.g., QUIC connection, etc.). See excerpt(s) above and below, for example (emphasis added, if any): "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
67	The apparatus of claim 65, wherein the apparatus is configured such that the first type of time period is started in response to the received packet.	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 65 and is configured such that the first type of time period (e.g., idle-type timeout, etc.) is started in response to the received packet. See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle_timeout parameter that is detected by a recipient of such packet. "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
		"7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability

Value	Parameter Name	Specification
0x0000	original_connection_id	Section 18.1
0x0001	idle_timeout	Section 18.1
0x0002	stateless_reset_token	Section 18.1
0x0003	max_packet_size	Section 18.1
0x0004	initial_max_data	Section 18.1
0x0005	initial_max_stream_data_bidi_local	Section 18.1
0x0006	initial_max_stream_data_bidi_remote	Section 18.1
0x0007	initial_max_stream_data_uni	Section 18.1
0x0008	initial_max_streams_bidi	Section 18.1
0x0009	initial_max_streams_uni	Section 18.1
0x000a	ack_delay_exponent	Section 18.1
0x000b	max_ack_delay	Section 18.1
0x000c	disable_migration	Section 18.1
0x000d	preferred_address	Section 18.1
0x000e	active_connection_id_limit	Section 18.1

Table 6: Initial QUIC Transport Parameters Entries

QUIC: A UDP-Based Multiplexed and Secure Transport https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
68	The apparatus of claim 65, wherein the apparatus is configured such that the first type of time period is started in response to the sent packet.	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 65 and is configured such that the first type of time period (e.g., idle-type timeout, etc.) is started in response to the sent packet. See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle_timeout parameter that is detected by a recipient of such packet. "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
		"7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
		+--------+--------------------------------------+---------------+ | Value | Parameter Name | Specification | +--------+--------------------------------------+---------------+ | 0x0000 | original_connection_id | Section 18.1 | | 0x0001 | idle_timeout | Section 18.1 | | 0x0002 | stateless_reset_token | Section 18.1 | | 0x0003 | max_packet_size | Section 18.1 | | 0x0004 | initial_max_data | Section 18.1 | | 0x0005 | initial_max_stream_data_bidi_local | Section 18.1 | | 0x0006 | initial_max_stream_data_bidi_remote | Section 18.1 | | 0x0007 | initial_max_stream_data_uni | Section 18.1 | | 0x0008 | initial_max_streams_bidi | Section 18.1 | | 0x0009 | initial_max_streams_uni | Section 18.1 | | 0x000a | ack_delay_exponent | Section 18.1 | | 0x000b | max_ack_delay | Section 18.1 | | 0x000c | disable_migration | Section 18.1 | | 0x000d | preferred_address | Section 18.1 | | 0x000e | active_connection_id_limit | Section 18.1 | +--------+--------------------------------------+---------------+ Table 6: Initial QUIC Transport Parameters Entries *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
69	The apparatus of claim 65, wherein the apparatus is	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 65 and is configured such that, during the second time period

Claim	Claim Elements	Applicability
	configured such that, during the second time period of the first type of time period, an empty packet is sent without causing a timer utilized to detect the second time period to be reset.	(e.g., idle timeout in connection with 1-RTT packets, etc.) of the first type of time period (e.g., idle-type timeout, etc.), an empty packet is sent (i.e. a packet carries no application data, etc.) without causing a timer utilized to detect the second time period to be reset. See excerpt(s) above and below, for example (emphasis added, if any): **Note**: Based upon information and belief, keep-alive signals are not required as a result of using idle timers and reducing the number of keep-alive signals to zero and, without a negotiated idle timeout, neither endpoint of a connection knows when the other side might timeout, so keep-alives (e.g., PINGs) have to be sent at unnecessarily shorter intervals to insure the connection to stay active. Further, with a negotiated idle time period, only one side sends one or more keep-alives, if a keep-alive is used. The QUIC IETF transport spec recognizes the need to avoid unnecessary keep-alives (e.g., PINGs). See section 10.1.2 of the QUIC transport draft at https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-34#section-10.1.2 and section 5.1 of the IETF HTTP/3 draft at https://tools.ietf.org/id/draft-ietf-quic-http-34.html#name-idle-connections. "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity.

Claim	Claim Elements	Applicability
		The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "19.2. PING Frame Endpoints can use PING frames (type=0x01) to verify that their peers are still alive or to check reachability to the peer. The PING frame contains no additional fields. The receiver of a PING frame simply needs to acknowledge the packet containing this frame. The PING frame can be used to keep a connection alive when an application or application protocol wishes to prevent the connection from timing out. An application protocol SHOULD provide guidance about the conditions under which generating a PING is recommended. This guidance SHOULD indicate whether it is the client or the server that is expected to send the PING. Having both endpoints send PING frames without coordination can produce an excessive number of packets and poor performance. A connection will time out if no packets are sent or received for a period longer than the time negotiated using the max_idle_timeout transport parameter (see Section 10). However, state in middleboxes might time out earlier than that. Though REQ-5 in [RFC4787] recommends a 2 minute timeout interval, experience shows that sending packets every 15 to 30 seconds is necessary to prevent the majority of middleboxes from losing state for UDP flows." https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-27#section-19.2

Claim	Claim Elements	Applicability
71	The apparatus of claim 65, wherein the apparatus is configured such that the third duration is determined using a second parameter field that is communicated in the first non-TCP connection and that identifies second metadata.	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 65 and is configured such that the third duration (e.g., duration associated with probe timeout, etc.) is determined using a second parameter field (e.g., second idle_timeout parameter field, etc.) that is communicated in the first non-TCP connection (e.g., QUIC connection, etc.) and that identifies second metadata (e.g., second idle timeout value, etc.). See excerpt(s) above and below, for example (emphasis added, if any): "18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298].

Claim	Claim Elements	Applicability
		6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "10.2. Idle Timeout

Claim	Claim Elements	Applicability
		If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily modified based on the value received in the idle_timeout field of the connection negotiation packet. "idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
72	The apparatus of claim 71, wherein the apparatus is configured such that the second duration is determined using the first parameter field identifying the first metadata, and the second parameter field identifying the second metadata.	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 71 and is configured such that the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.) is determined using the first parameter field (e.g., first idle_timeout parameter field, etc.) identifying the first metadata (e.g., first idle timeout value, etc.), and the second parameter field (e.g., second idle_timeout parameter field, etc.) identifying the second metadata (e.g., second idle timeout value, etc.). See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling. 7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake.

Claim	Claim Elements	Applicability
		The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original connection id, preferred address, stateless reset token, ack delay exponent and active connection id limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial_max_data and either initial_max_streams_bidi and

Claim	Claim Elements	Applicability
		initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity.

Claim	Claim Elements	Applicability
		The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "QUIC provides reliable, ordered delivery of the cryptographic handshake data. QUIC packet protection is used to encrypt as much of the handshake protocol as possible. The cryptographic handshake MUST provide the following properties: • authenticated key exchange, where ○ a server is always authenticated, ○ a client is optionally authenticated, ○ every connection produces distinct and unrelated keys, ○ keying material is usable for packet protection for both 0-RTT and 1-RTT packets, and ○ 1-RTT keys have forward secrecy • authenticated values for the transport parameters of the peer (see Section 7.3) • authenticated negotiation of an application protocol (TLS uses ALPN [RFC7301] for this purpose)" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily

Claim	Claim Elements	Applicability
		modified based on the value received in the idle_timeout field of the connection negotiation packet. "idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO

Claim	Claim Elements	Applicability
		When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3

Claim	Claim Elements	Applicability
73	The apparatus of claim 65, wherein the apparatus is configured such that: the first non-TCP packet is not a synchronize (SYN) packet;	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 65 and is configured such that: the first non-TCP packet (e.g., first QUIC negotiation packet, etc.) is not a synchronize (SYN) packet; See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle_timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
		+--------+--------------------------------------+---------------+ | Value | Parameter Name | Specification | +--------+--------------------------------------+---------------+ | 0x0000 | original_connection_id | Section 18.1 | | 0x0001 | idle_timeout | Section 18.1 | | 0x0002 | stateless_reset_token | Section 18.1 | | 0x0003 | max_packet_size | Section 18.1 | | 0x0004 | initial_max_data | Section 18.1 | | 0x0005 | initial_max_stream_data_bidi_local | Section 18.1 | | 0x0006 | initial_max_stream_data_bidi_remote | Section 18.1 | | 0x0007 | initial_max_stream_data_uni | Section 18.1 | | 0x0008 | initial_max_streams_bidi | Section 18.1 | | 0x0009 | initial_max_streams_uni | Section 18.1 | | 0x000a | ack_delay_exponent | Section 18.1 | | 0x000b | max_ack_delay | Section 18.1 | | 0x000c | disable_migration | Section 18.1 | | 0x000d | preferred_address | Section 18.1 | | 0x000e | active_connection_id_limit | Section 18.1 | +--------+--------------------------------------+---------------+ Table 6: Initial QUIC Transport Parameters Entries *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 **Note**: The following serves to distinguish the use of SYN handshake packets from QUIC negotiation packets where parameter values like idle_timeout may be exchanged.

Claim	Claim Elements	Applicability
		"The synchronization requires each side to send its own initial sequence number and to receive a confirmation of it in acknowledgment from the other side. Each side must also receive the other side's initial sequence number and send a confirming acknowledgment. 1) A --> B SYN my sequence number is X 2) A <-- B ACK your sequence number is X 3) A <-- B SYN my sequence number is Y 4) A --> B ACK your sequence number is Y Because steps 2 and 3 can be combined in a single message this is called the three way (or three message) handshake. A three way handshake is necessary because sequence numbers are not tied to a global clock in the network, and TCPs may have different mechanisms for picking the ISN's. The receiver of the first SYN has no way of knowing whether the segment was an old delayed one or not, unless it remembers the last sequence number used on the connection (which is not always possible), and so it must ask the sender to verify this SYN. The three way handshake and the advantages of a clock-driven scheme are discussed in [3]." "Request for Comments" (RFC) document RFC 793 (September 1981) https://tools.ietf.org/html/rfc793
	the first non-TCP connection is not a TCP extension connection that involves a TCP extension;	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 65 and is configured wherein the first non-TCP connection (e.g., QUIC connection, etc.) is not a TCP extension connection that involves a TCP extension; See excerpt(s) above and below, for example (emphasis added, if any): **Note**: The following evidence is related to and/or describes the QUIC standard contained herein.

Claim	Claim Elements	Applicability
		"On the surface, QUIC is very similar to TCP+TLS+HTTP/2 implemented on UDP. ...However, since QUIC is built on top of UDP, it suffers from no such limitations." https://www.chromium.org/quic
	the first type of time period includes an idle time period that does not include a user timeout period;	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 65 and is configured wherein the first type of time period (e.g., idle-type timeout, etc.) includes an idle time period that does not include a user timeout period; See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle_timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability

Value	Parameter Name	Specification
0x0000	original_connection_id	Section 18.1
0x0001	idle_timeout	Section 18.1
0x0002	stateless_reset_token	Section 18.1
0x0003	max_packet_size	Section 18.1
0x0004	initial_max_data	Section 18.1
0x0005	initial_max_stream_data_bidi_local	Section 18.1
0x0006	initial_max_stream_data_bidi_remote	Section 18.1
0x0007	initial_max_stream_data_uni	Section 18.1
0x0008	initial_max_streams_bidi	Section 18.1
0x0009	initial_max_streams_uni	Section 18.1
0x000a	ack_delay_exponent	Section 18.1
0x000b	max_ack_delay	Section 18.1
0x000c	disable_migration	Section 18.1
0x000d	preferred_address	Section 18.1
0x000e	active_connection_id_limit	Section 18.1

Table 6: Initial QUIC Transport Parameters Entries

QUIC: A UDP-Based Multiplexed and Secure Transport https://tools.ietf.org/html/draft-ietf-quic-transport-22

"18. Transport Parameter Encoding

Claim	Claim Elements	Applicability
		The format of the transport parameters is the TransportParameters struct from Figure 15. This is described using the presentation language from Section 3 of [TLS13]. enum { original_connection_id(0), idle_timeout(1), stateless_reset_token(2), max_packet_size(3), initial_max_data(4), initial_max_stream_data_bidi_local(5), initial_max_stream_data_bidi_remote(6), initial_max_stream_data_uni(7), initial_max_streams_bidi(8), initial_max_streams_uni(9), ack_delay_exponent(10), max_ack_delay(11), disable_migration(12), preferred_address(13), active_connection_id_limit(14), (65535) } TransportParameterId;" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily modified based on the value received in the idle_timeout field of the connection negotiation packet.

Claim	Claim Elements	Applicability
		"idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	the first time period is only capable of being detected during the first non-TCP connection set up;	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 65 and is configured wherein the first time period (e.g., idle timeout in connection with 0-RTT packets, etc.) is only capable of being detected during the first non-TCP connection (e.g., QUIC connection, etc.) set up; See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling. 7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake.

Claim	Claim Elements	Applicability
		The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and active_connection_id_limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes

Claim	Claim Elements	Applicability
		initial max data and either initial max streams bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	the first duration is not negotiated between the first and second nodes for the first non-TCP connection using (a-1), (a-2), nor (a-3), and the second duration and the third duration are negotiated between the first and second nodes for the first non-TCP connection using at least	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 65 and is configured wherein the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.) is not negotiated between the first and second nodes for the first non-TCP connection (e.g., QUIC connection, etc.) using (a-1), (a-2), nor (a-3), and the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.) and the third duration (e.g., duration associated with probe timeout, etc.) are negotiated between the first and second nodes for the first non-TCP connection using at least one of (a-1), (a-2), or (a-3); See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle timeout parameter that is detected by a recipient of such packet.

Claim	Claim Elements	Applicability
	one of (a-1), (a-2), or (a-3);	"7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling. 7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and active_connection_id_limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data.

Claim	Claim Elements	Applicability
		If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial max data and either initial max streams bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL VIOLATION.”

Claim	Claim Elements	Applicability
		QUIC: A UDP-Based Multiplexed and Secure Transport https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC

Claim	Claim Elements	Applicability
		implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time”

Claim	Claim Elements	Applicability
		QUIC Loss Detection and Congestion Control https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "QUIC provides reliable, ordered delivery of the cryptographic handshake data. QUIC packet protection is used to encrypt as much of the handshake protocol as possible. The cryptographic handshake MUST provide the following properties: • authenticated key exchange, where ○ a server is always authenticated, ○ a client is optionally authenticated, ○ every connection produces distinct and unrelated keys, ○ keying material is usable for packet protection for both 0-RTT and 1-RTT packets, and ○ 1-RTT keys have forward secrecy • authenticated values for the transport parameters of the peer (see Section 7.3) • authenticated negotiation of an application protocol (TLS uses ALPN [RFC7301] for this purpose)" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	the first duration is used and the second duration is not used, for detecting the first type of time period until the first duration is changed to the second duration;	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 65 and is configured wherein the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.) is used and the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.) is not used, for detecting the first type of time period (e.g., idle-type timeout, etc.) until the first duration is changed to the second duration; See excerpt(s) above and below, for example (emphasis added, if any):

Claim	Claim Elements	Applicability
		Note: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily detected based on the value received in the idle_timeout field of the connection negotiation packet. "idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10. Connection Termination An established QUIC connection can be terminated in one of three ways: • idle timeout (Section 10.2) • immediate close (Section 10.3) • stateless reset (Section 10.4) An endpoint MAY discard connection state if it does not have a validated path on which it can send packets (see Section 8.2)." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an

Claim	Claim Elements	Applicability
		ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and active connection id limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value.

Claim	Claim Elements	Applicability
		A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial max data and either initial max streams bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's

Claim	Claim Elements	Applicability
		updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	the second duration is determined based on a first algorithm, and the third duration is determined based on a second algorithm that is different from the first algorithm;	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 65 and is configured wherein the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.) is determined based on a first algorithm (e.g., used to negotiate an idle timeout period between nodes, etc.), and the third duration (e.g., duration associated with probe timeout, etc.) is determined based on a second algorithm (e.g., used to determine the probe timeout, etc.) that is different from the first algorithm; See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
		"18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3.

Claim	Claim Elements	Applicability
		The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity.

Claim	Claim Elements	Applicability
		The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 **Note**: Based upon information and belief, keep-alive signals are not required as a result of using idle timers and reducing the number of keep-alive signals to zero and, without a negotiated idle timeout, neither endpoint of a connection knows when the other side might timeout, so keep-alives (e.g., PINGs) have to be sent at unnecessarily shorter intervals to insure the connection to stay active. Further, with a negotiated idle time period, only one side sends one or more keep-alives, if a keep-alive is used. The QUIC IETF transport spec recognizes the need to avoid unnecessary keep-alives (e.g., PINGs). See section 10.1.2 of the QUIC transport draft at https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-34#section-10.1.2 and section 5.1 of the IETF HTTP/3 draft at https://tools.ietf.org/id/draft-ietf-quic-http-34.html#name-idle-connections. "19.2. PING Frame Endpoints can use PING frames (type=0x01) to verify that their peers are still alive or to check reachability to the peer. The PING frame contains no additional fields. The receiver of a PING frame simply needs to acknowledge the packet containing this frame.

Claim	Claim Elements	Applicability
		The PING frame can be used to keep a connection alive when an application or application protocol wishes to prevent the connection from timing out. An application protocol SHOULD provide guidance about the conditions under which generating a PING is recommended. This guidance SHOULD indicate whether it is the client or the server that is expected to send the PING. **Having both endpoints send PING frames without coordination can produce an excessive number of packets and poor performance.** **A connection will time out if no packets are sent or received for a period longer than the time negotiated using the max_idle_timeout transport parameter** (see Section 10). However, state in middleboxes might time out earlier than that. Though REQ-5 in [RFC4787] recommends a 2 minute timeout interval, experience shows that sending packets every 15 to 30 seconds is necessary to prevent the majority of middleboxes from losing state for UDP flows." https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-27#section-19.2
	the third time period is detected only in response to no acknowledgement packet being received in the first non-TCP connection and processed as an acknowledgement, during the third duration, for a sent packet sent in the first non-TCP connection;	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 65 and is configured wherein the third time period (e.g., probe timeout, etc.) is detected only in response to no acknowledgement packet being received in the first non-TCP connection (e.g., QUIC connection, etc.) and processed as an acknowledgement, during the third duration (e.g., duration associated with probe timeout, etc.), for a sent packet sent in the first non-TCP connection; See excerpt(s) above and below, for example (emphasis added, if any): "18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to

Claim	Claim Elements	Applicability
		1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement.

Claim	Claim Elements	Applicability
		The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a

Claim	Claim Elements	Applicability
		Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	during the at least portion of the first non-TCP connection including the at least portion of the first non-TCP connection set up, the detection of the first time period of the first type of time period is based on the first duration, by being only based on whether any non-TCP packet is received for the first non-TCP connection set up during the first duration;	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 65 and is configured wherein, during the at least portion of the first non-TCP connection (e.g., QUIC connection, etc.) including the at least portion of the first non-TCP connection set up, the detection of the first time period (e.g., idle timeout in connection with 0-RTT packets, etc.) of the first type of time period (e.g., idle-type timeout, etc.) is based on the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.), by being only based on whether any non-TCP packet is received for the first non-TCP connection set up during the first duration; See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling. 7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for

Claim	Claim Elements	Applicability
		transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original connection id, preferred address, stateless reset token, ack delay exponent and active connection id limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni

Claim	Claim Elements	Applicability
		Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial max data and either initial max streams bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "A connection will time out if no packets are sent or received for a period longer than the time negotiated using the max_idle_timeout transport parameter (see Section 10). However, state in middleboxes might time out earlier than that. Though REQ-5 in [RFC4787] recommends a 2 minute timeout interval, experience shows that sending packets every 15 to 30 seconds is necessary to prevent the majority of middleboxes from losing state for UDP flows." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/id/draft-ietf-quic-transport-27.txt

Claim	Claim Elements	Applicability
	after the first duration is changed to the second duration, the detection of the second time period of the first type of time period is based on the second duration, by being only based on whether, during the second duration, no packet is received and processed to keep the first non-TCP connection at least partially active, nor is expected to be received in the first non-TCP connection; and	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 65 and is configured wherein, after the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.) is changed to the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.), the detection of the second time period (e.g., idle timeout in connection with 1-RTT packets, etc.) of the first type of time period (e.g., idle-type timeout, etc.) is based on the second duration, by being only based on whether, during the second duration, no packet is received (e.g., connection remains idle, etc.) and processed to keep the first non-TCP connection (e.g., QUIC connection, etc.) at least partially active, nor is expected to be received in the first non-TCP connection; and See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily detected based on the value received in the idle_timeout field of the connection negotiation packet. "idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10. Connection Termination An established QUIC connection can be terminated in one of three ways: • idle timeout (Section 10.2) • immediate close (Section 10.3) • stateless reset (Section 10.4)

Claim	Claim Elements	Applicability
		An endpoint MAY discard connection state if it does not have a validated path on which it can send packets (see Section 8.2)." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "A connection will time out if no packets are sent or received for a period longer than the time negotiated using the max_idle_timeout transport parameter (see Section 10). However, state in middleboxes might time out earlier than that. Though REQ-5 in [RFC4787]

Claim	Claim Elements	Applicability
		recommends a 2 minute timeout interval, experience shows that sending packets every 15 to 30 seconds is necessary to prevent the majority of middleboxes from losing state for UDP flows." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/id/draft-ietf-quic-transport-27.txt "7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and active_connection_id_limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data.

Claim	Claim Elements	Applicability
		If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial max data and either initial max streams bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL VIOLATION.”

Claim	Claim Elements	Applicability
		QUIC: A UDP-Based Multiplexed and Secure Transport https://tools.ietf.org/html/draft-ietf-quic-transport-22
	an idle timeout attribute is utilized for the first type of time period, and a user timeout attribute, that is separate from the idle timeout attribute, is utilized for controlling communication of one or more acknowledgment packets.	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 65 and is configured wherein an idle timeout attribute is utilized for the first type of time period (e.g., idle-type timeout, etc.), and a user timeout attribute (e.g., acknowledgement in the form of PING frames, etc.), that is separate from the idle timeout attribute, is utilized for controlling communication of one or more acknowledgment packets. See excerpt(s) above and below, for example (emphasis added, if any): "19.2. PING Frame Endpoints can use PING frames (type=0x01) to verify that their peers are still alive or to check reachability to the peer. The PING frame contains no additional fields. The receiver of a PING frame simply needs to acknowledge the packet containing this frame. The PING frame can be used to keep a connection alive when an application or application protocol wishes to prevent the connection from timing out. An application protocol SHOULD provide guidance about the conditions under which generating a PING is recommended. This guidance SHOULD indicate whether it is the client or the server that is expected to send the PING. Having both endpoints send PING frames without coordination can produce an excessive number of packets and poor performance. A connection will time out if no packets are sent or received for a period longer than the time negotiated using the max_idle_timeout transport parameter (see Section 10). However, state in middleboxes might time out earlier than that. Though REQ-5 in [RFC4787] recommends a 2 minute timeout interval, experience shows that sending packets every 15 to 30 seconds is necessary to prevent the majority of middleboxes from losing state for UDP flows."

Claim	Claim Elements	Applicability
		https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-27#section-19.2
74	The apparatus of claim 73, wherein the apparatus is configured such that: the third time period is detected utilizing a timer that is not utilized to detect either of the first and second time periods; and	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 73 and is configured such that: the third time period (e.g., probe timeout, etc.) is detected utilizing a timer that is not utilized to detect either of the first and second time periods; and See excerpt(s) above and below, for example (emphasis added, if any): "18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO

Claim	Claim Elements	Applicability
		When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "10.2. Idle Timeout

Claim	Claim Elements	Applicability
		If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	the second duration is the same as the first duration, despite being capable of being different from the first duration.	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) that infringes claim 73 and is configured wherein the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.) is the same as the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.), despite being capable of being different from the first duration. See excerpt(s) above and below, for example (emphasis added, if any): "10.2. Idle Timeout

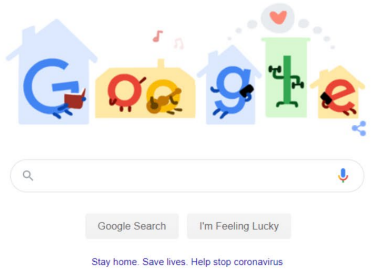
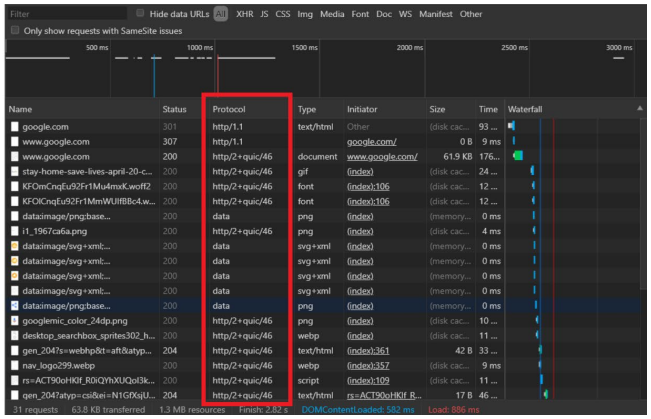
Claim	Claim Elements	Applicability
		If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout **can** be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake.

Claim	Claim Elements	Applicability
		The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original connection id, preferred address, stateless reset token, ack delay exponent and active connection id limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial_max_data and either initial_max_streams_bidi and

Claim	Claim Elements	Applicability
		initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
75	A non-transitory computer readable storage medium storing one or more programs, the one or more programs comprising instructions, which when executed by a first node, cause the first node to:	Google, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform with a non-transitory computer readable storage medium storing one or more programs (e.g. HTML pages, etc.), the one or more programs comprising instructions (e.g., server software, etc.), which when executed by a first node (e.g., one of a QUIC-compliant server or client, etc.), cause the first node to: See excerpt(s) above and below, for example (emphasis added, if any): **Note**: Google admits that their "Google Cloud Platform" uses QUIC: "Google Cloud Platform Blog... we're happy to be the first major public cloud to offer QUIC support for our HTTPS load balancers."

Claim	Claim Elements	Applicability
		https://cloudplatform.googleblog.com/2018/06/Introducing-QUIC-support-for-HTTPS-load-balancing.html **Note**: The "Google Cloud Platform" that, as established above, uses QUIC, is also used for ALL of Google's services: "Google Cloud Platform (GCP), offered by Google (company), is a suite of cloud computing services that runs on the same infrastructure that Google uses internally for its end-user products, such as Google Search, Gmail, file storage, and YouTube" https://en.wikipedia.org/wiki/Google_Cloud_Platform **Note**: At least a portion of the citations herein are made to the QUIC standard found at: https://tools.ietf.org/html/draft-ietf-quic-transport-22, and/or https://tools.ietf.org/html/draft-ietf-quic-transport-27. Based upon information and belief, the pertinent portions of such version of the QUIC standard may also be found in one or more other versions. For completeness, all of the versions below (and future iterations thereof) are hereby incorporated by reference: https://tools.ietf.org/html/draft-ietf-quic-transport-00 https://tools.ietf.org/html/draft-ietf-quic-transport-01 https://tools.ietf.org/html/draft-ietf-quic-transport-02 https://tools.ietf.org/html/draft-ietf-quic-transport-03 https://tools.ietf.org/html/draft-ietf-quic-transport-04 https://tools.ietf.org/html/draft-ietf-quic-transport-05 https://tools.ietf.org/html/draft-ietf-quic-transport-06 https://tools.ietf.org/html/draft-ietf-quic-transport-07 https://tools.ietf.org/html/draft-ietf-quic-transport-08 https://tools.ietf.org/html/draft-ietf-quic-transport-09 https://tools.ietf.org/html/draft-ietf-quic-transport-10 https://tools.ietf.org/html/draft-ietf-quic-transport-11 https://tools.ietf.org/html/draft-ietf-quic-transport-12

Claim	Claim Elements	Applicability
		https://tools.ietf.org/html/draft-ietf-quic-transport-13 https://tools.ietf.org/html/draft-ietf-quic-transport-14 https://tools.ietf.org/html/draft-ietf-quic-transport-15 https://tools.ietf.org/html/draft-ietf-quic-transport-16 https://tools.ietf.org/html/draft-ietf-quic-transport-17 https://tools.ietf.org/html/draft-ietf-quic-transport-18 https://tools.ietf.org/html/draft-ietf-quic-transport-19 https://tools.ietf.org/html/draft-ietf-quic-transport-20 https://tools.ietf.org/html/draft-ietf-quic-transport-21 https://tools.ietf.org/html/draft-ietf-quic-transport-22 https://tools.ietf.org/html/draft-ietf-quic-transport-23 https://tools.ietf.org/html/draft-ietf-quic-transport-24 https://tools.ietf.org/html/draft-ietf-quic-transport-25 https://tools.ietf.org/html/draft-ietf-quic-transport-26 https://tools.ietf.org/html/draft-ietf-quic-transport-27 **Note:** Though these contentions may refer to a specific version of the QUIC standard, the presented evidence is not exclusive to this specific version and is compliant with and/or supported by current versions of the QUIC standard, including QUIC_VERSION_IETF_RFC_V1 (https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-34), QUIC_VERSION_IETF_DRAFT_25 (https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-25), QUIC_VERSION_IETF_DRAFT_27 (https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-27), QUIC_VERSION_IETF_DRAFT_28 (https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-28), QUIC_VERSION_IETF_DRAFT_29 (https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-29), and/or QUIC_VERSION_50 which relies in part on IETF_QUIC_INVARIANTS_06 (https://tools.ietf.org/html/draft-ietf-quic-invariants-06) and/or IETF_QUIC_TRANSPORT_17 (https://tools.ietf.org/html/draft-ietf-quic-transport-17), each of which, upon information and belief, has been utilized by Google LLC through its accused instrumentalities.

Claim	Claim Elements	Applicability
		Note: Below is a web page of Google (https://www.google.com/).  
	receive first information on which at least a first duration for detecting a first type of time period is based;	Google, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform with a non-transitory computer readable storage medium is configured to receive first information (e.g., first value, etc.) on which at least a first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.) for detecting a first type of time period (e.g., idle-type timeout, etc.) is based; See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily detected based on the value received in the idle_ timeout field of the connection negotiation packet. "idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled."

Claim	Claim Elements	Applicability
		QUIC: A UDP-Based Multiplexed and Secure Transport https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10. Connection Termination An established QUIC connection can be terminated in one of three ways: • idle timeout (Section 10.2) • immediate close (Section 10.3) • stateless reset (Section 10.4) An endpoint MAY discard connection state if it does not have a validated path on which it can send packets (see Section 8.2)." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a

Claim	Claim Elements	Applicability
		Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original connection id, preferred address, stateless reset token, ack delay exponent and active connection id limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that

Claim	Claim Elements	Applicability
		accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial max data and either initial max streams bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
		"3.3. Session resumption versus Keep-alive Because QUIC is encapsulated in UDP, applications using QUIC must deal with short idle timeouts. Deployed stateful middleboxes will generally establish state for UDP flows on the first packet state, and keep state for much shorter idle periods than for TCP. According to a 2010 study ([Hatonen10]), UDP applications can assume that any NAT binding or other state entry will be expired after just thirty seconds of inactivity. A QUIC application has three strategies to deal with this issue: • Ignore it, if the application-layer protocol consists only of interactions with no or very short idle periods. • Ensure there are no long idle periods. • Resume the session after a long idle period, using 0-RTT resumption when appropriate. ... Alternatively, the client (but not the server) can use session resumption instead of sending keepalive traffic. In this case, a client that wants to send data to a server over a connection idle longer than the server's idle timeout (available from the idle timeout transport parameter) can simply reconnect. When possible, this reconnection can use 0-RTT session resumption, reducing the latency involved with restarting the connection. This of course only applies in cases in which 0-RTT data is safe, when the client is the restarting peer, and when the data to be sent is idempotent." https://www.ietf.org/id/draft-ietf-quic-applicability-07.txt
	identify a first resource for a first connection, where the first connection does not utilize Transmission Control Protocol (TCP);	Google, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform with a non-transitory computer readable storage medium is configured to identify a first resource (e.g. a storage and/or data structure for storing connection IDs, a connection state, send/recv buffers, etc.) for a first connection (e.g., QUIC

Claim	Claim Elements	Applicability
		connection, etc.), where the first connection does not utilize Transmission Control Protocol (TCP); See excerpt(s) above and below, for example (emphasis added, if any): "Google Cloud Platform (GCP), offered by Google (company), is a suite of cloud computing services that runs on the same infrastructure that Google uses internally for its end-user products, such as Google Search, Gmail, file storage, and YouTube" https://en.wikipedia.org/wiki/Google_Cloud_Platform
	generate a first packet including a first parameter field identifying first metadata for use in determining a second duration for detecting the first type of time period;	Google, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform with a non-transitory computer readable storage medium is configured to generate a first packet (e.g., QUIC negotiation packet, etc.) including a first parameter field (e.g., first idle_timeout parameter field, etc.) identifying first metadata (e.g., first idle timeout value, etc.) for use in determining a second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.) for detecting the first type of time period (e.g., idle-type timeout, etc.); See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle_timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability

Value	Parameter Name	Specification
0x0000	original_connection_id	Section 18.1
0x0001	idle_timeout	Section 18.1
0x0002	stateless_reset_token	Section 18.1
0x0003	max_packet_size	Section 18.1
0x0004	initial_max_data	Section 18.1
0x0005	initial_max_stream_data_bidi_local	Section 18.1
0x0006	initial_max_stream_data_bidi_remote	Section 18.1
0x0007	initial_max_stream_data_uni	Section 18.1
0x0008	initial_max_streams_bidi	Section 18.1
0x0009	initial_max_streams_uni	Section 18.1
0x000a	ack_delay_exponent	Section 18.1
0x000b	max_ack_delay	Section 18.1
0x000c	disable_migration	Section 18.1
0x000d	preferred_address	Section 18.1
0x000e	active_connection_id_limit	Section 18.1

Table 6: Initial QUIC Transport Parameters Entries

QUIC: A UDP-Based Multiplexed and Secure Transport https://tools.ietf.org/html/draft-ietf-quic-transport-22

"18. Transport Parameter Encoding

Claim	Claim Elements	Applicability
		The format of the transport parameters is the TransportParameters struct from Figure 15. This is described using the presentation language from Section 3 of [TLS13]. enum { original_connection_id(0), idle_timeout(1), stateless_reset_token(2), max_packet_size(3), initial_max_data(4), initial_max_stream_data_bidi_local(5), initial_max_stream_data_bidi_remote(6), initial_max_stream_data_uni(7), initial_max_streams_bidi(8), initial_max_streams_uni(9), ack_delay_exponent(10), max_ack_delay(11), disable_migration(12), preferred_address(13), active_connection_id_limit(14), (65535) } TransportParameterId;" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily modified based on the value received in the idle_timeout field of the connection negotiation packet.

Claim	Claim Elements	Applicability
		"idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "QUIC provides reliable, ordered delivery of the cryptographic handshake data. QUIC packet protection is used to encrypt as much of the handshake protocol as possible. The cryptographic handshake MUST provide the following properties: • authenticated key exchange, where ○ a server is always authenticated, ○ a client is optionally authenticated, ○ every connection produces distinct and unrelated keys, ○ keying material is usable for packet protection for both 0-RTT and 1-RTT packets, and ○ 1-RTT keys have forward secrecy • authenticated values for the transport parameters of the peer (see Section 7.3) • authenticated negotiation of an application protocol (TLS uses ALPN [RFC7301] for this purpose)" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	set up the first connection, by sending, from the first node to a second node, the first packet to provide the first metadata to the second node, for use by	Google, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform with a non-transitory computer readable storage medium is configured to set up the first connection (e.g., QUIC connection, etc.), by sending, from the first node (e.g., one of a QUIC-compliant server or client, etc.) to a second node (e.g., the other one of the QUIC-compliant server or client, etc.), the first packet (e.g., QUIC negotiation packet, etc.) to provide the first metadata (e.g., first idle timeout value, etc.) to the

Claim	Claim Elements	Applicability
	the second node in determining the second duration for detecting the first type of time period;	second node, for use by the second node in determining the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.) for detecting the first type of time period (e.g., idle-type timeout, etc.); See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle_timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability

Value	Parameter Name	Specification
0x0000	original_connection_id	Section 18.1
0x0001	idle_timeout	Section 18.1
0x0002	stateless_reset_token	Section 18.1
0x0003	max_packet_size	Section 18.1
0x0004	initial_max_data	Section 18.1
0x0005	initial_max_stream_data_bidi_local	Section 18.1
0x0006	initial_max_stream_data_bidi_remote	Section 18.1
0x0007	initial_max_stream_data_uni	Section 18.1
0x0008	initial_max_streams_bidi	Section 18.1
0x0009	initial_max_streams_uni	Section 18.1
0x000a	ack_delay_exponent	Section 18.1
0x000b	max_ack_delay	Section 18.1
0x000c	disable_migration	Section 18.1
0x000d	preferred_address	Section 18.1
0x000e	active_connection_id_limit	Section 18.1

Table 6: Initial QUIC Transport Parameters Entries

QUIC: A UDP-Based Multiplexed and Secure Transport https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
		"QUIC provides reliable, ordered delivery of the cryptographic handshake data. QUIC packet protection is used to encrypt as much of the handshake protocol as possible. The cryptographic handshake MUST provide the following properties: • authenticated key exchange, where ○ a server is always authenticated, ○ a client is optionally authenticated, ○ every connection produces distinct and unrelated keys, ○ keying material is usable for packet protection for both 0-RTT and 1-RTT packets, and ○ 1-RTT keys have forward secrecy • authenticated values for the transport parameters of the peer (see Section 7.3) • authenticated negotiation of an application protocol (TLS uses ALPN [RFC7301] for this purpose)" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	in response to detecting, based on the first duration and by the first node during at least a portion of the first connection including at least a portion of the first connection set up, a first time period of the first type of time period, at least partially close the first connection, by releasing, by the first	Google, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform with a non-transitory computer readable storage medium is configured to, in response to detecting, based on the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.) and by the first node (e.g., one of a QUIC-compliant server or client, etc.) during at least a portion of the first connection (e.g., QUIC connection, etc.) including at least a portion of the first connection set up, a first time period (e.g., idle timeout in connection with 0-RTT packets, etc.) of the first type of time period (e.g., idle-type timeout, etc.), at least partially close (e.g., terminate, etc.) the first connection, by releasing, by the first node, the first resource (e.g. a storage and/or data structure for storing connection IDs, a connection state, send/recv buffers, etc.) for the first connection;

Claim	Claim Elements	Applicability
	node, the first resource for the first connection;	See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily detected based on the value received in the idle_timeout field of the connection negotiation packet. "idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10. Connection Termination An established QUIC connection can be terminated in one of three ways: • idle timeout (Section 10.2) • immediate close (Section 10.3) • stateless reset (Section 10.4) An endpoint MAY discard connection state if it does not have a validated path on which it can send packets (see Section 8.2)." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO)." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
		"7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and active_connection_id_limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local

Claim	Claim Elements	Applicability
		• initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial max data and either initial max streams bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	in response to detecting, based on the second duration and by the first node after the first	Google, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform with a non-transitory computer readable storage medium is configured to, in response to detecting, based on the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.) and by the first node

Claim	Claim Elements	Applicability
	duration is changed to the second duration, a second time period of the first type of time period, at least partially close the first connection, by releasing, by the first node, the first resource for the first connection; and	(e.g., one of a QUIC-compliant server or client, etc.) after the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.) is changed to the second duration, a second time period (e.g., idle timeout in connection with 1-RTT packets, etc.) of the first type of time period (e.g., idle-type timeout, etc.), at least partially close (e.g., terminate, etc.) the first connection (e.g., QUIC connection, etc.), by releasing, by the first node, the first resource (e.g. a storage and/or data structure for storing connection IDs, a connection state, send/recv buffers, etc.) for the first connection; and See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily detected based on the value received in the idle_timeout field of the connection negotiation packet. "idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10. Connection Termination An established QUIC connection can be terminated in one of three ways: • idle timeout (Section 10.2) • immediate close (Section 10.3) • stateless reset (Section 10.4) An endpoint MAY discard connection state if it does not have a validated path on which it can send packets (see Section 8.2)." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
		"10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT

Claim	Claim Elements	Applicability
		packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and active_connection_id_limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni

Claim	Claim Elements	Applicability
		Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial_max_data and either initial_max_streams_bidi and initial_max_stream_data_bidi_remote, or initial_max_streams_uni and initial_max_stream_data_uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	in response to detecting, based on a third duration and by the first node, a third time period, at least partially close the first connection, by releasing, by the first node, the first resource allocated for the first connection, the third duration being determined	Google, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform with a non-transitory computer readable storage medium is configured to, in response to detecting, based on a third duration (e.g., duration associated with probe timeout, etc.) and by the first node (e.g., one of a QUIC-compliant server or client, etc.), a third time period (e.g., probe timeout, etc.), at least partially close (e.g., terminate, etc.) the first connection (e.g., QUIC connection, etc.), by releasing, by the first node, the first resource (e.g. a storage and/or data structure for storing connection IDs, a connection state, send/recv buffers, etc.) allocated for the first connection, the third duration being determined based on a first algorithm (e.g., used to determine the probe timeout, etc.) that is different from a second algorithm (e.g., used to negotiate an idle timeout period

Claim	Claim Elements	Applicability
	based on a first algorithm that is different from a second algorithm on which a determination of the second duration is based.	between nodes, etc.) on which a determination of the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.) is based. See excerpt(s) above and below, for example (emphasis added, if any): "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "18.1 Transport Parameter Definitions …

Claim	Claim Elements	Applicability
		• max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt),

Claim	Claim Elements	Applicability
		the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3
76	The non-transitory computer readable storage medium of claim 75, wherein the instructions, when executed by the first node, cause the first node to: in response to detecting, by the first node, the third time period that is a second type of time period including an acknowledgment timeout period during which no acknowledgement packet	Google, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform with a non-transitory computer readable storage medium that infringes claim 75 and is configured wherein the instructions (e.g., server software, etc.), when executed by the first node (e.g., one of a QUIC-compliant server or client, etc.), cause the first node to: in response to detecting, by the first node, the third time period (e.g., probe timeout, etc.) that is a second type of time period (e.g., probe-type timeout, etc.) including an acknowledgment timeout period during which no acknowledgement packet is received in the first connection (e.g., QUIC connection, etc.) and processed as an acknowledgement for a sent packet sent in the first connection: See excerpt(s) above and below, for example (emphasis added, if any): "18.1 Transport Parameter Definitions …

Claim	Claim Elements	Applicability
	is received in the first connection and processed as an acknowledgement for a sent packet sent in the first connection:	• max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt),

Claim	Claim Elements	Applicability
		the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity.

Claim	Claim Elements	Applicability
		The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	release, by the first node, the first resource for the first connection, wherein:	Google, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform with a non-transitory computer readable storage medium that infringes claim 75 and is configured to release, by the first node (e.g., one of a QUIC-compliant server or client, etc.), the first resource (e.g. a storage and/or data structure for storing connection IDs, a connection state, send/recv buffers, etc.) for the first connection (e.g., QUIC connection, etc.), wherein: See excerpt(s) above and below, for example (emphasis added, if any): "10. Connection Termination An established QUIC connection can be terminated in one of three ways: • idle timeout (Section 10.2) • immediate close (Section 10.3) • stateless reset (Section 10.4) An endpoint MAY discard connection state if it does not have a validated path on which it can send packets (see Section 8.2)." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout

Claim	Claim Elements	Applicability
		If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO)." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	the first time period and the second time period are started in response to at least one of a received packet or a sent packet; and	Google, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform with a non-transitory computer readable storage medium that infringes claim 75 and is configured wherein the first time period (e.g., idle timeout in connection with 0-RTT packets, etc.) and the second time period (e.g., idle timeout in connection with 1-RTT packets, etc.) are started in response to at least one of a received packet or a sent packet; and See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "18.1 Transport Parameter Definitions …

Claim	Claim Elements	Applicability
		• max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt),

Claim	Claim Elements	Applicability
		the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity.

Claim	Claim Elements	Applicability
		The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	during the at least portion of the first connection including the at least portion of the first connection set up, the detection of the first time period of the first type of time period is based on the first duration, by being based on whether, during the first duration after the first time period is started, no packet is received and processed to keep the first connection at least partially active; and	Google, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform with a non-transitory computer readable storage medium that infringes claim 75 and is configured wherein, during the at least portion of the first connection (e.g., QUIC connection, etc.) including the at least portion of the first connection set up, the detection of the first time period (e.g., idle timeout in connection with 0-RTT packets, etc.) of the first type of time period (e.g., idle-type timeout, etc.) is based on the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.), by being based on whether, during the first duration after the first time period is started, no packet is received (e.g., connection remains idle, etc.) and processed to keep the first connection at least partially active; and See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints

Claim	Claim Elements	Applicability
		are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling. 7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and active_connection_id_limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters.

Claim	Claim Elements	Applicability
		• initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial max data and either initial max streams bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout

Claim	Claim Elements	Applicability
		If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	based on the first duration being changed to the second duration, the detection of the second time period of the first type of time period is based on the second duration, by being based on whether, during the second duration after the	Google, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform with a non-transitory computer readable storage medium that infringes claim 75 and is configured wherein, based on the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.) being changed to the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.), the detection of the second time period (e.g., idle timeout in connection with 1-RTT packets, etc.) of the first type of time period (e.g., idle-type timeout, etc.) is based on the second duration, by being based on whether, during the second duration after the second time period is started, no packet is received (e.g., connection remains idle, etc.) and processed to keep the first connection (e.g., QUIC connection, etc.) at least partially active.

Claim	Claim Elements	Applicability
	second time period is started, no packet is received and processed to keep the first connection at least partially active.	See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily detected based on the value received in the idle_timeout field of the connection negotiation packet. "idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling. 7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake.

Claim	Claim Elements	Applicability
		The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original connection id, preferred address, stateless reset token, ack delay exponent and active connection id limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial_max_data and either initial_max_streams_bidi and

Claim	Claim Elements	Applicability
		initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10. Connection Termination An established QUIC connection can be terminated in one of three ways: • idle timeout (Section 10.2) • immediate close (Section 10.3) • stateless reset (Section 10.4) An endpoint MAY discard connection state if it does not have a validated path on which it can send packets (see Section 8.2)." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout

Claim	Claim Elements	Applicability
		If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "QUIC provides reliable, ordered delivery of the cryptographic handshake data. QUIC packet protection is used to encrypt as much of the handshake protocol as possible. The cryptographic handshake MUST provide the following properties: • authenticated key exchange, where ○ a server is always authenticated, ○ a client is optionally authenticated, ○ every connection produces distinct and unrelated keys,

Claim	Claim Elements	Applicability
		○ keying material is usable for packet protection for both 0-RTT and 1-RTT packets, and ○ 1-RTT keys have forward secrecy • authenticated values for the transport parameters of the peer (see Section 7.3) • authenticated negotiation of an application protocol (TLS uses ALPN [RFC7301] for this purpose)" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
77	The non-transitory computer readable storage medium of claim 76, wherein the first connection set up further includes, in addition to (a-1) sending the first packet that is also for use in determining the third duration of the third time period:	Google, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform with a non-transitory computer readable storage medium that infringes claim 76 and is configured wherein the first connection (e.g., QUIC connection, etc.) set up further includes, in addition to (a-1) sending the first packet that is also for use in determining the third duration (e.g., duration associated with probe timeout, etc.) of the third time period (e.g., probe timeout, etc.): See excerpt(s) above and below, for example (emphasis added, if any): "18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
		"6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion.

Claim	Claim Elements	Applicability
		A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
	(a-2) after sending the first packet and without any other communication between the first node and the second node after (a-1) and before (a-2), receiving, at the first node from the second node, a second packet, and (a-3) after receiving the second packet and without any other communication between the first node and the second node after (a-2) and before (a-3), sending, from the first node to the second node, a third packet, and further wherein:	Google, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform with a non-transitory computer readable storage medium that infringes claim 76 and is configured wherein, (a-2) after sending the first packet and without any other communication between the first node (e.g., one of a QUIC-compliant server or client, etc.) and the second node (e.g., the other one of the QUIC-compliant server or client, etc.) after (a-1) and before (a-2), receiving, at the first node from the second node, a second packet, and (a-3) after receiving the second packet and without any other communication between the first node and the second node after (a-2) and before (a-3), sending, from the first node to the second node, a third packet, and further wherein: See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle_timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
		+--------+--------------------------------------+---------------+ | Value | Parameter Name | Specification | +--------+--------------------------------------+---------------+ | 0x0000 | original_connection_id | Section 18.1 | | 0x0001 | idle_timeout | Section 18.1 | | 0x0002 | stateless_reset_token | Section 18.1 | | 0x0003 | max_packet_size | Section 18.1 | | 0x0004 | initial_max_data | Section 18.1 | | 0x0005 | initial_max_stream_data_bidi_local | Section 18.1 | | 0x0006 | initial_max_stream_data_bidi_remote | Section 18.1 | | 0x0007 | initial_max_stream_data_uni | Section 18.1 | | 0x0008 | initial_max_streams_bidi | Section 18.1 | | 0x0009 | initial_max_streams_uni | Section 18.1 | | 0x000a | ack_delay_exponent | Section 18.1 | | 0x000b | max_ack_delay | Section 18.1 | | 0x000c | disable_migration | Section 18.1 | | 0x000d | preferred_address | Section 18.1 | | 0x000e | active_connection_id_limit | Section 18.1 | +--------+--------------------------------------+---------------+ Table 6: Initial QUIC Transport Parameters Entries *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	the first duration, the second duration, and the	Google, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform with a non-transitory computer readable storage medium

Claim	Claim Elements	Applicability
	third duration are capable of being of different durations;	that infringes claim 76 and is configured wherein the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.), the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.), and the third duration (e.g., duration associated with probe timeout, etc.) are capable of being of different durations; See excerpt(s) above and below, for example (emphasis added, if any): "The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process.

Claim	Claim Elements	Applicability
		A client MUST NOT use remembered values for the following parameters: original connection id, preferred address, stateless reset token, ack delay exponent and active connection id limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial max data and either initial max streams bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni.

Claim	Claim Elements	Applicability
		A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "QUIC provides reliable, ordered delivery of the cryptographic handshake data. QUIC packet protection is used to encrypt as much of the handshake protocol as possible. The cryptographic handshake MUST provide the following properties: • authenticated key exchange, where ○ a server is always authenticated, ○ a client is optionally authenticated, ○ every connection produces distinct and unrelated keys, ○ keying material is usable for packet protection for both 0-RTT and 1-RTT packets, and ○ 1-RTT keys have forward secrecy • authenticated values for the transport parameters of the peer (see Section 7.3) • authenticated negotiation of an application protocol (TLS uses ALPN [RFC7301] for this purpose)"

Claim	Claim Elements	Applicability
		QUIC: A UDP-Based Multiplexed and Secure Transport https://tools.ietf.org/html/draft-ietf-quic-transport-22
	the second time period and the third time period are capable of being detected during the first connection set up and thereafter during the first connection;	Google, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform with a non-transitory computer readable storage medium that infringes claim 76 and is configured wherein the second time period (e.g., idle timeout in connection with 1-RTT packets, etc.) and the third time period (e.g., probe timeout, etc.) are capable of being detected during the first connection (e.g., QUIC connection, etc.) set up and thereafter during the first connection; See excerpt(s) above and below, for example (emphasis added, if any): "18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and

Claim	Claim Elements	Applicability
		F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3

Claim	Claim Elements	Applicability
		"10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	the first packet is not a synchronize (SYN) packet;	Google, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform with a non-transitory computer readable storage medium that infringes claim 76 and is configured wherein the first packet is not a synchronize (SYN) packet; See excerpt(s) above and below, for example (emphasis added, if any):

Claim	Claim Elements	Applicability
		Note: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle_timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability

Value	Parameter Name	Specification
0x0000	original_connection_id	Section 18.1
0x0001	idle_timeout	Section 18.1
0x0002	stateless_reset_token	Section 18.1
0x0003	max_packet_size	Section 18.1
0x0004	initial_max_data	Section 18.1
0x0005	initial_max_stream_data_bidi_local	Section 18.1
0x0006	initial_max_stream_data_bidi_remote	Section 18.1
0x0007	initial_max_stream_data_uni	Section 18.1
0x0008	initial_max_streams_bidi	Section 18.1
0x0009	initial_max_streams_uni	Section 18.1
0x000a	ack_delay_exponent	Section 18.1
0x000b	max_ack_delay	Section 18.1
0x000c	disable_migration	Section 18.1
0x000d	preferred_address	Section 18.1
0x000e	active_connection_id_limit	Section 18.1

Table 6: Initial QUIC Transport Parameters Entries

QUIC: A UDP-Based Multiplexed and Secure Transport https://tools.ietf.org/html/draft-ietf-quic-transport-22

Note: The following serves to distinguish the use of SYN handshake packets from QUIC negotiation packets where parameter values like idle_timeout may be exchanged.

Claim	Claim Elements	Applicability
		"The synchronization requires each side to send its own initial sequence number and to receive a confirmation of it in acknowledgment from the other side. Each side must also receive the other side's initial sequence number and send a confirming acknowledgment. 1) A --> B SYN my sequence number is X 2) A <-- B ACK your sequence number is X 3) A <-- B SYN my sequence number is Y 4) A --> B ACK your sequence number is Y Because steps 2 and 3 can be combined in a single message this is called the three way (or three message) handshake. A three way handshake is necessary because sequence numbers are not tied to a global clock in the network, and TCPs may have different mechanisms for picking the ISN's. The receiver of the first SYN has no way of knowing whether the segment was an old delayed one or not, unless it remembers the last sequence number used on the connection (which is not always possible), and so it must ask the sender to verify this SYN. The three way handshake and the advantages of a clock-driven scheme are discussed in [3]." "Request for Comments" (RFC) document RFC 793 (September 1981) https://tools.ietf.org/html/rfc793
	the first connection does not utilize a TCP extension;	Google, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform with a non-transitory computer readable storage medium that infringes claim 76 and is configured wherein the first connection (e.g., QUIC connection, etc.) does not utilize a TCP extension; See excerpt(s) above and below, for example (emphasis added, if any): **Note**: The following evidence is related to and/or describes the QUIC standard contained herein.

Claim	Claim Elements	Applicability
		"On the surface, QUIC is very similar to TCP+TLS+HTTP/2 implemented on UDP. ...However, since QUIC is built on top of UDP, it suffers from no such limitations." https://www.chromium.org/quic
	the first type of time period includes an idle time period that does not include a user timeout period;	Google, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform with a non-transitory computer readable storage medium that infringes claim 76 and is configured wherein the first type of time period (e.g., idle-type timeout, etc.) includes an idle time period that does not include a user timeout period; See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle_timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability

Value	Parameter Name	Specification
0x0000	original_connection_id	Section 18.1
0x0001	idle_timeout	Section 18.1
0x0002	stateless_reset_token	Section 18.1
0x0003	max_packet_size	Section 18.1
0x0004	initial_max_data	Section 18.1
0x0005	initial_max_stream_data_bidi_local	Section 18.1
0x0006	initial_max_stream_data_bidi_remote	Section 18.1
0x0007	initial_max_stream_data_uni	Section 18.1
0x0008	initial_max_streams_bidi	Section 18.1
0x0009	initial_max_streams_uni	Section 18.1
0x000a	ack_delay_exponent	Section 18.1
0x000b	max_ack_delay	Section 18.1
0x000c	disable_migration	Section 18.1
0x000d	preferred_address	Section 18.1
0x000e	active_connection_id_limit	Section 18.1

Table 6: Initial QUIC Transport Parameters Entries

QUIC: A UDP-Based Multiplexed and Secure Transport https://tools.ietf.org/html/draft-ietf-quic-transport-22

"18. Transport Parameter Encoding

Claim	Claim Elements	Applicability
		The format of the transport parameters is the TransportParameters struct from Figure 15. This is described using the presentation language from Section 3 of [TLS13]. enum { original_connection_id(0), idle_timeout(1), stateless_reset_token(2), max_packet_size(3), initial_max_data(4), initial_max_stream_data_bidi_local(5), initial_max_stream_data_bidi_remote(6), initial_max_stream_data_uni(7), initial_max_streams_bidi(8), initial_max_streams_uni(9), ack_delay_exponent(10), max_ack_delay(11), disable_migration(12), preferred_address(13), active_connection_id_limit(14), (65535) } TransportParameterId;" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily modified based on the value received in the idle_timeout field of the connection negotiation packet.

Claim	Claim Elements	Applicability
		"idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	the first time period is only capable of being detected during the first connection set up;	Google, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform with a non-transitory computer readable storage medium that infringes claim 76 and is configured wherein the first time period (e.g., idle timeout in connection with 0-RTT packets, etc.) is only capable of being detected during the first connection (e.g., QUIC connection, etc.) set up; See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling. 7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT

Claim	Claim Elements	Applicability
		packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and active_connection_id limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni

Claim	Claim Elements	Applicability
		Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial_max_data and either initial_max_streams_bidi and initial_max_stream_data_bidi_remote, or initial_max_streams_uni and initial_max_stream_data_uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	the first duration is not negotiated between the first and second nodes for the first connection using (a-1), (a-2), nor (a-3), and the second duration and the third duration are negotiated between the first and second nodes for the first connection using	Google, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform with a non-transitory computer readable storage medium that infringes claim 76 and is configured wherein the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.) is not negotiated between the first and second nodes for the first connection (e.g., QUIC connection, etc.) using (a-1), (a-2), nor (a-3), and the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.) and the third duration (e.g., duration associated with probe timeout, etc.) are negotiated between the first and second nodes for the first connection using at least one of (a-1), (a-2), or (a-3);

Claim	Claim Elements	Applicability
	at least one of (a-1), (a-2), or (a-3);	See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling. 7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original connection id, preferred address, stateless reset token, ack delay exponent and active connection id limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value.

Claim	Claim Elements	Applicability
		A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial max data and either initial max streams bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of

Claim	Claim Elements	Applicability
		transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
		"6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion.

Claim	Claim Elements	Applicability
		A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "QUIC provides reliable, ordered delivery of the cryptographic handshake data. QUIC packet protection is used to encrypt as much of the handshake protocol as possible. The cryptographic handshake MUST provide the following properties: • authenticated key exchange, where ○ a server is always authenticated, ○ a client is optionally authenticated, ○ every connection produces distinct and unrelated keys, ○ keying material is usable for packet protection for both 0-RTT and 1-RTT packets, and ○ 1-RTT keys have forward secrecy • authenticated values for the transport parameters of the peer (see Section 7.3) • authenticated negotiation of an application protocol (TLS uses ALPN [RFC7301] for this purpose)" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	the first duration is used and the second duration is not used, for detecting the first type of time period until the first duration is	Google, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform with a non-transitory computer readable storage medium that infringes claim 76 and is configured wherein the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.) is used and the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.) is not

Claim	Claim Elements	Applicability
	changed to the second duration;	used, for detecting the first type of time period (e.g., idle-type timeout, etc.) until the first duration is changed to the second duration; See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily detected based on the value received in the idle_timeout field of the connection negotiation packet. "idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10. Connection Termination An established QUIC connection can be terminated in one of three ways: • idle timeout (Section 10.2) • immediate close (Section 10.3) • stateless reset (Section 10.4) An endpoint MAY discard connection state if it does not have a validated path on which it can send packets (see Section 8.2)." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout

Claim	Claim Elements	Applicability
		If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake.

Claim	Claim Elements	Applicability
		The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original connection id, preferred address, stateless reset token, ack delay exponent and active connection id limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial_max_data and either initial_max_streams_bidi and

Claim	Claim Elements	Applicability
		initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	the second duration is determined based on a first algorithm, and the third duration is determined based on a second algorithm that is different from the first algorithm;	Google, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform with a non-transitory computer readable storage medium that infringes claim 76 and is configured wherein the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.) is determined based on a first algorithm (e.g., used to negotiate an idle timeout period between nodes, etc.), and the third duration (e.g., duration associated with probe timeout, etc.) is determined based on a second algorithm (e.g., used to determine the probe timeout, etc.) that is different from the first algorithm; See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle timeout parameter that is detected by a recipient of such packet.

Claim	Claim Elements	Applicability
		"7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO

Claim	Claim Elements	Applicability
		When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "10.2. Idle Timeout

Claim	Claim Elements	Applicability
		If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 **Note**: Based upon information and belief, keep-alive signals are not required as a result of using idle timers and reducing the number of keep-alive signals to zero and, without a negotiated idle timeout, neither endpoint of a connection knows when the other side might timeout, so keep-alives (e.g., PINGs) have to be sent at unnecessarily shorter intervals to insure the connection to stay active. Further, with a negotiated idle time period, only one side sends one or more keep-alives, if a keep-alive is used. The QUIC IETF transport spec recognizes the need to avoid unnecessary keep-alives (e.g., PINGs). See section 10.1.2 of the QUIC transport draft at https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-34#section-10.1.2 and section

Claim	Claim Elements	Applicability
		5.1 of the IETF HTTP/3 draft at https://tools.ietf.org/id/draft-ietf-quic-http-34.html#name-idle-connections. "19.2. PING Frame Endpoints can use PING frames (type=0x01) to verify that their peers are still alive or to check reachability to the peer. The PING frame contains no additional fields. The receiver of a PING frame simply needs to acknowledge the packet containing this frame. The PING frame can be used to keep a connection alive when an application or application protocol wishes to prevent the connection from timing out. An application protocol SHOULD provide guidance about the conditions under which generating a PING is recommended. This guidance SHOULD indicate whether it is the client or the server that is expected to send the PING. **Having both endpoints send PING frames without coordination can produce an excessive number of packets and poor performance.** **A connection will time out if no packets are sent or received for a period longer than the time negotiated using the max_idle_timeout transport parameter** (see Section 10). However, state in middleboxes might time out earlier than that. Though REQ-5 in [RFC4787] recommends a 2 minute timeout interval, experience shows that sending packets every 15 to 30 seconds is necessary to prevent the majority of middleboxes from losing state for UDP flows." https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-27#section-19.2
	the third time period is detected only in response to no acknowledgement packet being received in the first connection and processed as an	Google, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform with a non-transitory computer readable storage medium that infringes claim 76 and is configured wherein the third time period (e.g., probe timeout, etc.) is detected only in response to no acknowledgement packet being received in the first connection (e.g., QUIC connection, etc.) and processed as an acknowledgement, during the

Claim	Claim Elements	Applicability
	acknowledgement, during the third duration, for a sent packet sent in the first connection;	third duration (e.g., duration associated with probe timeout, etc.), for a sent packet sent in the first connection; See excerpt(s) above and below, for example (emphasis added, if any): "18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows:

Claim	Claim Elements	Applicability
		PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO).

Claim	Claim Elements	Applicability
		Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	during the at least portion of the first connection including the at least portion of the first connection set up, the detection of the first time period of the first type of time period is based on the first duration, by being only based on whether any packet is received for the first connection set up during the first duration;	Google, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform with a non-transitory computer readable storage medium that infringes claim 76 and is configured wherein, during the at least portion of the first connection (e.g., QUIC connection, etc.) including the at least portion of the first connection set up, the detection of the first time period (e.g., idle timeout in connection with 0-RTT packets, etc.) of the first type of time period (e.g., idle-type timeout, etc.) is based on the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.), by being only based on whether any packet is received for the first connection set up during the first duration; See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle timeout parameter that is detected by a recipient of such packet.

Claim	Claim Elements	Applicability
		"7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and active connection id limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-

Claim	Claim Elements	Applicability
		protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial max data and either initial max streams bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if

Claim	Claim Elements	Applicability
		those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	based on the first duration being changed to the second duration, the detection of the second time period of the first type of time period is based on the second duration, by being only based on whether, during the second duration, no packet is received and processed to keep the first connection at least partially active, nor is expected to be received in the first connection; and	Google, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform with a non-transitory computer readable storage medium that infringes claim 76 and is configured wherein, based on the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.) being changed to the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.), the detection of the second time period (e.g., idle timeout in connection with 1-RTT packets, etc.) of the first type of time period (e.g., idle-type timeout, etc.) is based on the second duration, by being only based on whether, during the second duration, no packet is received (e.g., connection remains idle, etc.) and processed to keep the first connection (e.g., QUIC connection, etc.) at least partially active, nor is expected to be received in the first connection; and See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily detected based on the value received in the idle_timeout field of the connection negotiation packet. "idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
		"7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling. 7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and active_connection_id_limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data.

Claim	Claim Elements	Applicability
		If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial max data and either initial max streams bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL VIOLATION.”

Claim	Claim Elements	Applicability
		QUIC: A UDP-Based Multiplexed and Secure Transport https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10. Connection Termination An established QUIC connection can be terminated in one of three ways: • idle timeout (Section 10.2) • immediate close (Section 10.3) • stateless reset (Section 10.4) An endpoint MAY discard connection state if it does not have a validated path on which it can send packets (see Section 8.2)." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a

Claim	Claim Elements	Applicability
		Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "QUIC provides reliable, ordered delivery of the cryptographic handshake data. QUIC packet protection is used to encrypt as much of the handshake protocol as possible. The cryptographic handshake MUST provide the following properties: • authenticated key exchange, where ○ a server is always authenticated, ○ a client is optionally authenticated, ○ every connection produces distinct and unrelated keys, ○ keying material is usable for packet protection for both 0-RTT and 1-RTT packets, and ○ 1-RTT keys have forward secrecy • authenticated values for the transport parameters of the peer (see Section 7.3) • authenticated negotiation of an application protocol (TLS uses ALPN [RFC7301] for this purpose)" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	an idle timeout attribute is utilized for the first type of time period, and a user timeout attribute, that is separate from the idle timeout attribute, is	Google, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform with a non-transitory computer readable storage medium that infringes claim 76 and is configured wherein an idle timeout attribute is utilized for the first type of time period (e.g., idle-type timeout, etc.), and a user timeout attribute (e.g., acknowledgement in the form of PING frames, etc.), that is separate from the idle timeout attribute, is utilized for controlling communication of one or more acknowledgment packets.

Claim	Claim Elements	Applicability
	utilized for controlling communication of one or more acknowledgment packets.	See excerpt(s) above and below, for example (emphasis added, if any): "19.2. PING Frame Endpoints can use PING frames (type=0x01) to verify that their peers are still alive or to check reachability to the peer. The PING frame contains no additional fields. The receiver of a PING frame simply needs to acknowledge the packet containing this frame. The PING frame can be used to keep a connection alive when an application or application protocol wishes to prevent the connection from timing out. An application protocol SHOULD provide guidance about the conditions under which generating a PING is recommended. This guidance SHOULD indicate whether it is the client or the server that is expected to send the PING. Having both endpoints send PING frames without coordination can produce an excessive number of packets and poor performance. A connection will time out if no packets are sent or received for a period longer than the time negotiated using the max_idle_timeout transport parameter (see Section 10). However, state in middleboxes might time out earlier than that. Though REQ-5 in [RFC4787] recommends a 2 minute timeout interval, experience shows that sending packets every 15 to 30 seconds is necessary to prevent the majority of middleboxes from losing state for UDP flows." https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-27#section-19.2
78	A method, comprising: at at least a portion of a first node: identifying first information on which at least a first duration for detecting a first type of time period is based;	Google owns, controls, and/or uses at least one server computer, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to that performs a method including, at at least a portion of a first node (e.g., one of a QUIC-compliant server or client, etc.): identifying first information (e.g., first value, etc.) on which at least a first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.) for detecting a first type of time period (e.g., idle-type timeout, etc.) is based;

Claim	Claim Elements	Applicability
		See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily detected based on the value received in the idle_timeout field of the connection negotiation packet. "idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10. Connection Termination An established QUIC connection can be terminated in one of three ways: • idle timeout (Section 10.2) • immediate close (Section 10.3) • stateless reset (Section 10.4) An endpoint MAY discard connection state if it does not have a validated path on which it can send packets (see Section 8.2)." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO).

Claim	Claim Elements	Applicability
		Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process.

Claim	Claim Elements	Applicability
		A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and active_connection_id_limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial_max_data and either initial_max_streams_bidi and initial_max_stream_data_bidi_remote, or initial_max_streams_uni and initial_max_stream_data_uni.

Claim	Claim Elements	Applicability
		A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "3.3. Session resumption versus Keep-alive Because QUIC is encapsulated in UDP, applications using QUIC must deal with short idle timeouts. Deployed stateful middleboxes will generally establish state for UDP flows on the first packet state, and keep state for much shorter idle periods than for TCP. According to a 2010 study ([Hatonen10]), UDP applications can assume that any NAT binding or other state entry will be expired after just thirty seconds of inactivity. A QUIC application has three strategies to deal with this issue: • Ignore it, if the application-layer protocol consists only of interactions with no or very short idle periods. • Ensure there are no long idle periods. • Resume the session after a long idle period, using 0-RTT resumption when appropriate. ...

Claim	Claim Elements	Applicability
		Alternatively, the client (but not the server) can use session resumption instead of sending keepalive traffic. In this case, a client that wants to send data to a server over a connection idle longer than the server's idle timeout (available from the idle timeout transport parameter) can simply reconnect. When possible, this reconnection can use 0-RTT session resumption, reducing the latency involved with restarting the connection. This of course only applies in cases in which 0-RTT data is safe, when the client is the restarting peer, and when the data to be sent is idempotent." https://www.ietf.org/id/draft-ietf-quic-applicability-07.txt
	allocating a first resource for a first Transmission Control Protocol-variant (TCP-variant) connection;	Google owns, controls, and/or uses at least one server computer, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform for allocating a first resource (e.g. a storage and/or data structure for storing connection IDs, a connection state, send/recv buffers, etc.) for a first Transmission Control Protocol-variant (TCP-variant) connection (e.g., QUIC connection, etc.); See excerpt(s) above and below, for example (emphasis added, if any): "Google Cloud Platform (GCP), offered by Google (company), is a suite of cloud computing services that runs on the same infrastructure that Google uses internally for its end-user products, such as Google Search, Gmail, file storage, and YouTube" https://en.wikipedia.org/wiki/Google_Cloud_Platform
	generating a first TCP-variant packet including a first parameter field identifying first metadata for use in determining a second duration for detecting the first type of time period;	Google owns, controls, and/or uses at least one server computer, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform for generating a first TCP-variant packet including a first parameter field (e.g., first idle_timeout parameter field, etc.) identifying first metadata (e.g., first idle timeout value, etc.) for use in determining a second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.) for detecting the first type of time period (e.g., idle-type timeout, etc.); See excerpt(s) above and below, for example (emphasis added, if any):

Claim	Claim Elements	Applicability
		Note: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle_timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability

Value	Parameter Name	Specification
0x0000	original_connection_id	Section 18.1
0x0001	idle_timeout	Section 18.1
0x0002	stateless_reset_token	Section 18.1
0x0003	max_packet_size	Section 18.1
0x0004	initial_max_data	Section 18.1
0x0005	initial_max_stream_data_bidi_local	Section 18.1
0x0006	initial_max_stream_data_bidi_remote	Section 18.1
0x0007	initial_max_stream_data_uni	Section 18.1
0x0008	initial_max_streams_bidi	Section 18.1
0x0009	initial_max_streams_uni	Section 18.1
0x000a	ack_delay_exponent	Section 18.1
0x000b	max_ack_delay	Section 18.1
0x000c	disable_migration	Section 18.1
0x000d	preferred_address	Section 18.1
0x000e	active_connection_id_limit	Section 18.1

Table 6: Initial QUIC Transport Parameters Entries

QUIC: A UDP-Based Multiplexed and Secure Transport https://tools.ietf.org/html/draft-ietf-quic-transport-22

"18. Transport Parameter Encoding

Claim	Claim Elements	Applicability
		The format of the transport parameters is the TransportParameters struct from Figure 15. This is described using the presentation language from Section 3 of [TLS13]. enum { original_connection_id(0), idle_timeout(1), stateless_reset_token(2), max_packet_size(3), initial_max_data(4), initial_max_stream_data_bidi_local(5), initial_max_stream_data_bidi_remote(6), initial_max_stream_data_uni(7), initial_max_streams_bidi(8), initial_max_streams_uni(9), ack_delay_exponent(10), max_ack_delay(11), disable_migration(12), preferred_address(13), active_connection_id_limit(14), (65535) } TransportParameterId;" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily modified based on the value received in the idle_timeout field of the connection negotiation packet.

Claim	Claim Elements	Applicability
		"idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "QUIC provides reliable, ordered delivery of the cryptographic handshake data. QUIC packet protection is used to encrypt as much of the handshake protocol as possible. The cryptographic handshake MUST provide the following properties: • authenticated key exchange, where ○ a server is always authenticated, ○ a client is optionally authenticated, ○ every connection produces distinct and unrelated keys, ○ keying material is usable for packet protection for both 0-RTT and 1-RTT packets, and ○ 1-RTT keys have forward secrecy • authenticated values for the transport parameters of the peer (see Section 7.3) • authenticated negotiation of an application protocol (TLS uses ALPN [RFC7301] for this purpose)" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	setting up the first TCP-variant connection, by sending, from the first node to a second node, the first TCP-variant packet to provide the first metadata	Google owns, controls, and/or uses at least one server computer, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform for setting up the first TCP-variant connection (e.g., QUIC connection, etc.), by sending, from the first node (e.g., one of a QUIC-compliant server or client, etc.) to a second node (e.g., the other one of the QUIC-compliant server or client, etc.), the first TCP-variant packet to provide the first metadata (e.g., first idle timeout value, etc.) to the second node, for use by

Claim	Claim Elements	Applicability
	to the second node, for use by the second node in determining the second duration for detecting the first type of time period;	the second node in determining the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.) for detecting the first type of time period (e.g., idle-type timeout, etc.); See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle_timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability

Value	Parameter Name	Specification
0x0000	original_connection_id	Section 18.1
0x0001	idle_timeout	Section 18.1
0x0002	stateless_reset_token	Section 18.1
0x0003	max_packet_size	Section 18.1
0x0004	initial_max_data	Section 18.1
0x0005	initial_max_stream_data_bidi_local	Section 18.1
0x0006	initial_max_stream_data_bidi_remote	Section 18.1
0x0007	initial_max_stream_data_uni	Section 18.1
0x0008	initial_max_streams_bidi	Section 18.1
0x0009	initial_max_streams_uni	Section 18.1
0x000a	ack_delay_exponent	Section 18.1
0x000b	max_ack_delay	Section 18.1
0x000c	disable_migration	Section 18.1
0x000d	preferred_address	Section 18.1
0x000e	active_connection_id_limit	Section 18.1

Table 6: Initial QUIC Transport Parameters Entries

QUIC: A UDP-Based Multiplexed and Secure Transport https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
		"QUIC provides reliable, ordered delivery of the cryptographic handshake data. QUIC packet protection is used to encrypt as much of the handshake protocol as possible. The cryptographic handshake MUST provide the following properties: • authenticated key exchange, where ○ a server is always authenticated, ○ a client is optionally authenticated, ○ every connection produces distinct and unrelated keys, ○ keying material is usable for packet protection for both 0-RTT and 1-RTT packets, and ○ 1-RTT keys have forward secrecy • authenticated values for the transport parameters of the peer (see Section 7.3) • authenticated negotiation of an application protocol (TLS uses ALPN [RFC7301] for this purpose)" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	based on detecting, by the first node and during at least a portion of the first TCP-variant connection including at least a portion of the first TCP-variant connection set up, a first duration-based first time period of the first type of time period, at least partially closing the first TCP-variant connection,	Google owns, controls, and/or uses at least one server computer, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform for, based on detecting, by the first node (e.g., one of a QUIC-compliant server or client, etc.) and during at least a portion of the first TCP-variant connection (e.g., QUIC connection, etc.) including at least a portion of the first TCP-variant connection set up, a first duration-based first time period (e.g., idle timeout in connection with 0-RTT packets, etc.) of the first type of time period (e.g., idle-type timeout, etc.), at least partially closing the first TCP-variant connection, by releasing the first resource (e.g. a storage and/or data structure for storing connection IDs, a connection state, send/recv buffers, etc.) allocated for the first TCP-variant connection; See excerpt(s) above and below, for example (emphasis added, if any):

Claim	Claim Elements	Applicability
	by releasing the first resource allocated for the first TCP-variant connection;	**Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily detected based on the value received in the idle_timeout field of the connection negotiation packet. "idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10. Connection Termination An established QUIC connection can be terminated in one of three ways: • idle timeout (Section 10.2) • immediate close (Section 10.3) • stateless reset (Section 10.4) An endpoint MAY discard connection state if it does not have a validated path on which it can send packets (see Section 8.2)." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO)." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
		"7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original connection id, preferred address, stateless reset token, ack delay exponent and active connection id limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote

Claim	Claim Elements	Applicability
		• initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial_max_data and either initial_max_streams_bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	based on detecting, by the first node and based on the first duration being changed to the second duration, a second	Google owns, controls, and/or uses at least one server computer, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform for, based on detecting, by the first node (e.g., one of a QUIC-compliant server or client, etc.) and based on the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.) being changed to the second duration (e.g., duration associated with idle

Claim	Claim Elements	Applicability
	duration-based second time period of the first type of time period, at least partially closing the first TCP-variant connection, by releasing the first resource allocated for the first TCP-variant connection; and	timeout in connection with 1-RTT packets, etc.), a second duration-based second time period (e.g., idle timeout in connection with 1-RTT packets, etc.) of the first type of time period (e.g., idle-type timeout, etc.), at least partially closing the first TCP-variant connection (e.g., QUIC connection, etc.), by releasing the first resource (e.g. a storage and/or data structure for storing connection IDs, a connection state, send/recv buffers, etc.) allocated for the first TCP-variant connection; and See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily detected based on the value received in the idle_timeout field of the connection negotiation packet. "idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10. Connection Termination An established QUIC connection can be terminated in one of three ways: • idle timeout (Section 10.2) • immediate close (Section 10.3) • stateless reset (Section 10.4) An endpoint MAY discard connection state if it does not have a validated path on which it can send packets (see Section 8.2)." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
		"10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake.

Claim	Claim Elements	Applicability
		The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and active_connection_id_limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes

Claim	Claim Elements	Applicability
		initial max data and either initial max streams bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	based on detecting, by the first node and based on a third duration, a third duration-based third time period, at least partially closing the first TCP-variant connection, by releasing the first resource allocated for the first TCP-variant connection, where the third duration is determined based on a first algorithm that is different	Google owns, controls, and/or uses at least one server computer, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform for, based on detecting, by the first node (e.g., one of a QUIC-compliant server or client, etc.) and based on a third duration (e.g., duration associated with probe timeout, etc.), a third duration-based third time period (e.g., probe timeout, etc.), at least partially closing the first TCP-variant connection (e.g., QUIC connection, etc.), by releasing the first resource (e.g. a storage and/or data structure for storing connection IDs, a connection state, send/recv buffers, etc.) allocated for the first TCP-variant connection, where the third duration is determined based on a first algorithm (e.g., used to determine the probe timeout, etc.) that is different from a second algorithm (e.g., used to negotiate an idle timeout period between nodes, etc.) on which a determination of the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.) is based.

Claim	Claim Elements	Applicability
	from a second algorithm on which a determination of the second duration is based.	See excerpt(s) above and below, for example (emphasis added, if any): "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to

Claim	Claim Elements	Applicability
		1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement.

Claim	Claim Elements	Applicability
		The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3
80	The method of claim 78, wherein the first time period is only capable of being detected during the first TCP-variant connection set up.	Google infringes claim 78, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein the first time period (e.g., idle timeout in connection with 0-RTT packets, etc.) is only capable of being detected during the first TCP-variant connection (e.g., QUIC connection, etc.) set up. See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling.

Claim	Claim Elements	Applicability
		7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and active_connection_id_limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote

Claim	Claim Elements	Applicability
		• initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial_max_data and either initial_max_streams_bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
83	The method of claim 78, wherein the third time period is detected based on no acknowledgement packet being detected, for	Google infringes claim 78, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein the third time period (e.g., probe timeout, etc.) is detected based on no acknowledgement packet being detected, for the third duration (e.g., duration associated with probe timeout, etc.) that starts based on a sent packet being sent in the first TCP-variant connection (e.g., QUIC

Claim	Claim Elements	Applicability
	the third duration that starts based on a sent packet being sent in the first TCP-variant connection, as being received in the first TCP-variant connection and processed to acknowledge the sent packet.	connection, etc.), as being received in the first TCP-variant connection and processed to acknowledge the sent packet. See excerpt(s) above and below, for example (emphasis added, if any): "18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows:

Claim	Claim Elements	Applicability
		PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO).

Claim	Claim Elements	Applicability
		Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
84	The method of claim 83, wherein the first TCP-variant packet also includes data for use in determining the third time period.	Google infringes claim 83, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein the first TCP-variant packet (e.g., first QUIC negotiation packet, etc.) also includes data for use in determining the third time period (e.g., probe timeout, etc.). See excerpt(s) above and below, for example (emphasis added, if any): "18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to

Claim	Claim Elements	Applicability
		1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement.

Claim	Claim Elements	Applicability
		The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a

Claim	Claim Elements	Applicability
		Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
86	The method of claim 83, wherein the first duration is not negotiated between the first and second nodes during the first TCP-variant connection set up, the second duration is negotiated between the first and second nodes for the first TCP-variant connection, and the third duration of the third time period is negotiated between the first and second nodes for the first TCP-variant connection.	Google infringes claim 83, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.) is not negotiated between the first and second nodes during the first TCP-variant connection (e.g., QUIC connection, etc.) set up, the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.) is negotiated between the first and second nodes for the first TCP-variant connection, and the third duration (e.g., duration associated with probe timeout, etc.) of the third time period (e.g., probe timeout, etc.) is negotiated between the first and second nodes for the first TCP-variant connection. See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout

Claim	Claim Elements	Applicability
		If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake.

Claim	Claim Elements	Applicability
		The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original connection id, preferred address, stateless reset token, ack delay exponent and active connection id limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial_max_data and either initial_max_streams_bidi and

Claim	Claim Elements	Applicability
		initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
88	The method of claim 83, wherein the third time period is a second type of time period.	Google infringes claim 83, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein the third time period (e.g., probe timeout, etc.) is a second type of time period (e.g., probe-type timeout, etc.). See excerpt(s) above and below, for example (emphasis added, if any): "18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to

Claim	Claim Elements	Applicability
		1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement.

Claim	Claim Elements	Applicability
		The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a

Claim	Claim Elements	Applicability
		Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
103	The method of claim 78, wherein the first type of time period includes an idle time period.	Google infringes claim 78, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein the first type of time period (e.g., idle-type timeout, etc.) includes an idle time period. See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle_timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability

Value	Parameter Name	Specification
0x0000	original_connection_id	Section 18.1
0x0001	idle_timeout	Section 18.1
0x0002	stateless_reset_token	Section 18.1
0x0003	max_packet_size	Section 18.1
0x0004	initial_max_data	Section 18.1
0x0005	initial_max_stream_data_bidi_local	Section 18.1
0x0006	initial_max_stream_data_bidi_remote	Section 18.1
0x0007	initial_max_stream_data_uni	Section 18.1
0x0008	initial_max_streams_bidi	Section 18.1
0x0009	initial_max_streams_uni	Section 18.1
0x000a	ack_delay_exponent	Section 18.1
0x000b	max_ack_delay	Section 18.1
0x000c	disable_migration	Section 18.1
0x000d	preferred_address	Section 18.1
0x000e	active_connection_id_limit	Section 18.1

Table 6: Initial QUIC Transport Parameters Entries

QUIC: A UDP-Based Multiplexed and Secure Transport https://tools.ietf.org/html/draft-ietf-quic-transport-22

"18. Transport Parameter Encoding

Claim	Claim Elements	Applicability
		The format of the transport parameters is the TransportParameters struct from Figure 15. This is described using the presentation language from Section 3 of [TLS13]. enum { original_connection_id(0), idle_timeout(1), stateless_reset_token(2), max_packet_size(3), initial_max_data(4), initial_max_stream_data_bidi_local(5), initial_max_stream_data_bidi_remote(6), initial_max_stream_data_uni(7), initial_max_streams_bidi(8), initial_max_streams_uni(9), ack_delay_exponent(10), max_ack_delay(11), disable_migration(12), preferred_address(13), active_connection_id_limit(14), (65535) } TransportParameterId;" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily modified based on the value received in the idle_timeout field of the connection negotiation packet.

Claim	Claim Elements	Applicability
		"idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
104	The method of claim 103, wherein the idle time period is one: during which, no packet is communicated in the TCP-variant connection to keep the TCP-variant connection active.	Google infringes claim 103, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein the idle time period is one: during which, no packet is communicated in the TCP-variant connection (e.g., QUIC connection, etc.) to keep the TCP-variant connection active. See excerpt(s) above and below, for example (emphasis added, if any): "A connection will time out if no packets are sent or received for a period longer than the time negotiated using the max_idle_timeout transport parameter (see Section 10). However, state in middleboxes might time out earlier than that. Though REQ-5 in [RFC4787] recommends a 2 minute timeout interval, experience shows that sending packets every 15 to 30 seconds is necessary to prevent the majority of middleboxes from losing state for UDP flows." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/id/draft-ietf-quic-transport-27.txt
159	The method of claim 78, and further comprising: based on detecting, by the first node, the third time period that is a second type of time period including an acknowledgment timeout period during which no	Google infringes claim 78, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method further comprising: based on detecting, by the first node (e.g., one of a QUIC-compliant server or client, etc.), the third time period (e.g., probe timeout, etc.) that is a second type of time period (e.g., probe-type timeout, etc.) including an acknowledgment timeout period during which no acknowledgement packet is received in the first TCP-variant connection (e.g., QUIC connection, etc.) and processed as an acknowledgement for a sent packet sent in the first TCP-variant connection:

Claim	Claim Elements	Applicability
	acknowledgement packet is received in the first TCP-variant connection and processed as an acknowledgement for a sent packet sent in the first TCP-variant connection:	See excerpt(s) above and below, for example (emphasis added, if any): "18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay

Claim	Claim Elements	Applicability
		kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an

Claim	Claim Elements	Applicability
		ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	unilaterally at least partially closing the first TCP-variant connection, by releasing, by the first node, the first resource allocated for the first TCP-variant connection, wherein:	Google infringes claim 78, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method for unilaterally at least partially closing the first TCP-variant connection (e.g., QUIC connection, etc.), by releasing, by the first node (e.g., one of a QUIC-compliant server or client, etc.), the first resource (e.g. a storage and/or data structure for storing connection IDs, a connection state, send/recv buffers, etc.) allocated for the first TCP-variant connection, wherein: See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily detected based on the value received in the idle_timeout field of the connection negotiation packet. "idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled."

Claim	Claim Elements	Applicability
		QUIC: A UDP-Based Multiplexed and Secure Transport https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10. Connection Termination An established QUIC connection can be terminated in one of three ways: • idle timeout (Section 10.2) • immediate close (Section 10.3) • stateless reset (Section 10.4) An endpoint MAY discard connection state if it does not have a validated path on which it can send packets (see Section 8.2)." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a

Claim	Claim Elements	Applicability
		Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	the first TCP-variant connection set up further includes, in addition to (a-1) sending the first TCP-variant packet that is also for use in determining the third duration of the third time period:	Google infringes claim 78, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein the first TCP-variant connection (e.g., QUIC connection, etc.) set up further includes, in addition to (a-1) sending the first TCP-variant packet that is also for use in determining the third duration (e.g., duration associated with probe timeout, etc.) of the third time period (e.g., probe timeout, etc.): See excerpt(s) above and below, for example (emphasis added, if any): "18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a

Claim	Claim Elements	Applicability
		connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time”

Claim	Claim Elements	Applicability
		QUIC Loss Detection and Congestion Control https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	(a-2) after sending the first TCP-variant packet and without any other communication between the first node and the	Google infringes claim 78, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein, (a-2) after sending the first TCP-variant packet and without any other communication between the first node (e.g., one of a QUIC-compliant server or client, etc.) and the second node (e.g., the other one of the QUIC-compliant server or client, etc.) after (a-1) and before (a-2), receiving,

Claim	Claim Elements	Applicability
	second node after (a-1) and before (a-2), receiving, at the first node from the second node, a second TCP-variant packet, and (a-3) after receiving the second TCP-variant packet and without any other communication between the first node and the second node after (a-2) and before (a-3), sending, from the first node to the second node, a third TCP-variant packet, and further wherein:	at the first node from the second node, a second TCP-variant packet, and (a-3) after receiving the second TCP-variant packet and without any other communication between the first node and the second node after (a-2) and before (a-3), sending, from the first node to the second node, a third TCP-variant packet, and further wherein: See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle_timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
		+--------+--------------------------------------+---------------+ | Value | Parameter Name | Specification | +--------+--------------------------------------+---------------+ | 0x0000 | original_connection_id | Section 18.1 | | 0x0001 | idle_timeout | Section 18.1 | | 0x0002 | stateless_reset_token | Section 18.1 | | 0x0003 | max_packet_size | Section 18.1 | | 0x0004 | initial_max_data | Section 18.1 | | 0x0005 | initial_max_stream_data_bidi_local | Section 18.1 | | 0x0006 | initial_max_stream_data_bidi_remote | Section 18.1 | | 0x0007 | initial_max_stream_data_uni | Section 18.1 | | 0x0008 | initial_max_streams_bidi | Section 18.1 | | 0x0009 | initial_max_streams_uni | Section 18.1 | | 0x000a | ack_delay_exponent | Section 18.1 | | 0x000b | max_ack_delay | Section 18.1 | | 0x000c | disable_migration | Section 18.1 | | 0x000d | preferred_address | Section 18.1 | | 0x000e | active_connection_id_limit | Section 18.1 | +--------+--------------------------------------+---------------+ Table 6: Initial QUIC Transport Parameters Entries *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	the first duration, the second duration, and the	Google infringes claim 78, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein the first

Claim	Claim Elements	Applicability
	third duration are capable of being of different durations;	duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.), the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.), and the third duration (e.g., duration associated with probe timeout, etc.) are capable of being of different durations; See excerpt(s) above and below, for example (emphasis added, if any): "The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process.

Claim	Claim Elements	Applicability
		A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and active_connection_id_limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial_max_data and either initial_max_streams_bidi and initial_max_stream_data_bidi_remote, or initial_max_streams_uni and initial_max_stream_data_uni.

Claim	Claim Elements	Applicability
		A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "QUIC provides reliable, ordered delivery of the cryptographic handshake data. QUIC packet protection is used to encrypt as much of the handshake protocol as possible. The cryptographic handshake MUST provide the following properties: • authenticated key exchange, where ○ a server is always authenticated, ○ a client is optionally authenticated, ○ every connection produces distinct and unrelated keys, ○ keying material is usable for packet protection for both 0-RTT and 1-RTT packets, and ○ 1-RTT keys have forward secrecy • authenticated values for the transport parameters of the peer (see Section 7.3) • authenticated negotiation of an application protocol (TLS uses ALPN [RFC7301] for this purpose)"

Claim	Claim Elements	Applicability
		QUIC: A UDP-Based Multiplexed and Secure Transport https://tools.ietf.org/html/draft-ietf-quic-transport-22
	the second time period and the third time period are capable of being detected during the first TCP-variant connection set up and thereafter during the first TCP-variant connection; and	Google infringes claim 78, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein the second time period (e.g., idle timeout in connection with 1-RTT packets, etc.) and the third time period (e.g., probe timeout, etc.) are capable of being detected during the first TCP-variant connection (e.g., QUIC connection, etc.) set up and thereafter during the first TCP-variant connection; and See excerpt(s) above and below, for example (emphasis added, if any): "18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and

Claim	Claim Elements	Applicability
		F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3

Claim	Claim Elements	Applicability
		"10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	the first time period and the second time period are started based on at least one of a received packet or a sent packet; and	Google infringes claim 78, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein the first time period (e.g., idle timeout in connection with 0-RTT packets, etc.) and the second time period (e.g., idle timeout in connection with 1-RTT packets, etc.) are started based on at least one of a received packet or a sent packet; and See excerpt(s) above and below, for example (emphasis added, if any):

Claim	Claim Elements	Applicability
		Note: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and

Claim	Claim Elements	Applicability
		F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3

Claim	Claim Elements	Applicability
		"10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	during the at least portion of the first TCP-variant connection including the at least portion of the first TCP-variant connection set up, the detection of the first time period of the first	Google infringes claim 78, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein, during the at least portion of the first TCP-variant connection (e.g., QUIC connection, etc.) including the at least portion of the first TCP-variant connection set up, the detection of the first time period (e.g., idle timeout in connection with 0-RTT packets, etc.) of the first type of time period (e.g., idle-type timeout, etc.) is based on the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.), by being based on whether, during

Claim	Claim Elements	Applicability
	type of time period is based on the first duration, by being based on whether, during the first duration, no packet is received and processed to keep the first TCP-variant connection at least partially active; and	the first duration, no packet is received (e.g., connection remains idle, etc.) and processed to keep the first TCP-variant connection at least partially active; and See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling. 7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and

Claim	Claim Elements	Applicability
		active connection id limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial max data and either initial max streams bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported.

Claim	Claim Elements	Applicability
		When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried."

Claim	Claim Elements	Applicability
		QUIC: A UDP-Based Multiplexed and Secure Transport https://tools.ietf.org/html/draft-ietf-quic-transport-22
	based on the first duration being changed to the second duration, the detection of the second time period of the first type of time period is based on the second duration, by being based on whether, during the second duration, no packet is received and processed to keep the first TCP-variant connection at least partially active;	Google infringes claim 78, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein, based on the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.) being changed to the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.), the detection of the second time period (e.g., idle timeout in connection with 1-RTT packets, etc.) of the first type of time period (e.g., idle-type timeout, etc.) is based on the second duration, by being based on whether, during the second duration, no packet is received (e.g., connection remains idle, etc.) and processed to keep the first TCP-variant connection at least partially active; See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily detected based on the value received in the idle_timeout field of the connection negotiation packet. "idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints

Claim	Claim Elements	Applicability
		are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling. 7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and active_connection_id_limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters.

Claim	Claim Elements	Applicability
		• initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial max data and either initial max streams bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10. Connection Termination

Claim	Claim Elements	Applicability
		An established QUIC connection can be terminated in one of three ways: • idle timeout (Section 10.2) • immediate close (Section 10.3) • stateless reset (Section 10.4) An endpoint MAY discard connection state if it does not have a validated path on which it can send packets (see Section 8.2)." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
		"QUIC provides reliable, ordered delivery of the cryptographic handshake data. QUIC packet protection is used to encrypt as much of the handshake protocol as possible. The cryptographic handshake MUST provide the following properties: • authenticated key exchange, where ○ a server is always authenticated, ○ a client is optionally authenticated, ○ every connection produces distinct and unrelated keys, ○ keying material is usable for packet protection for both 0-RTT and 1-RTT packets, and ○ 1-RTT keys have forward secrecy • authenticated values for the transport parameters of the peer (see Section 7.3) • authenticated negotiation of an application protocol (TLS uses ALPN [RFC7301] for this purpose)" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	where only one of the first time period, the second time period, or the third time period is detected for the first TCP-variant connection.	Google infringes claim 78, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method where only one of the first time period (e.g., idle timeout in connection with 0-RTT packets, etc.), the second time period (e.g., idle timeout in connection with 1-RTT packets, etc.), or the third time period (e.g., probe timeout, etc.) is detected for the first TCP-variant connection (e.g., QUIC connection, etc.). See excerpt(s) above and below, for example (emphasis added, if any): "10.2. Idle Timeout

Claim	Claim Elements	Applicability
		If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
167	The method of claim 159, wherein: the first TCP-variant packet is not a synchronize (SYN) packet;	Google infringes claim 159, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein: the first TCP-variant packet is not a synchronize (SYN) packet; See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters

Claim	Claim Elements	Applicability
		During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
		+--------+--------------------------------------+---------------+ | Value | Parameter Name | Specification | +--------+--------------------------------------+---------------+ | 0x0000 | original_connection_id | Section 18.1 | | 0x0001 | idle_timeout | Section 18.1 | | 0x0002 | stateless_reset_token | Section 18.1 | | 0x0003 | max_packet_size | Section 18.1 | | 0x0004 | initial_max_data | Section 18.1 | | 0x0005 | initial_max_stream_data_bidi_local | Section 18.1 | | 0x0006 | initial_max_stream_data_bidi_remote | Section 18.1 | | 0x0007 | initial_max_stream_data_uni | Section 18.1 | | 0x0008 | initial_max_streams_bidi | Section 18.1 | | 0x0009 | initial_max_streams_uni | Section 18.1 | | 0x000a | ack_delay_exponent | Section 18.1 | | 0x000b | max_ack_delay | Section 18.1 | | 0x000c | disable_migration | Section 18.1 | | 0x000d | preferred_address | Section 18.1 | | 0x000e | active_connection_id_limit | Section 18.1 | +--------+--------------------------------------+---------------+ Table 6: Initial QUIC Transport Parameters Entries *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 **Note**: The following serves to distinguish the use of SYN handshake packets from QUIC negotiation packets where parameter values like idle_timeout may be exchanged.

Claim	Claim Elements	Applicability
		"The synchronization requires each side to send its own initial sequence number and to receive a confirmation of it in acknowledgment from the other side. Each side must also receive the other side's initial sequence number and send a confirming acknowledgment. 1) A --> B SYN my sequence number is X 2) A <-- B ACK your sequence number is X 3) A <-- B SYN my sequence number is Y 4) A --> B ACK your sequence number is Y Because steps 2 and 3 can be combined in a single message this is called the three way (or three message) handshake. A three way handshake is necessary because sequence numbers are not tied to a global clock in the network, and TCPs may have different mechanisms for picking the ISN's. The receiver of the first SYN has no way of knowing whether the segment was an old delayed one or not, unless it remembers the last sequence number used on the connection (which is not always possible), and so it must ask the sender to verify this SYN. The three way handshake and the advantages of a clock-driven scheme are discussed in [3]." "Request for Comments" (RFC) document RFC 793 (September 1981) https://tools.ietf.org/html/rfc793
	the first TCP-variant connection is not a TCP extension connection that involves a TCP extension;	Google infringes claim 159, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein the first TCP-variant connection (e.g., QUIC connection, etc.) is not a TCP extension connection that involves a TCP extension; See excerpt(s) above and below, for example (emphasis added, if any): **Note**: The following evidence is related to and/or describes the QUIC standard contained herein.

Claim	Claim Elements	Applicability
		"On the surface, QUIC is very similar to TCP+TLS+HTTP/2 implemented on UDP. ...However, since QUIC is built on top of UDP, it suffers from no such limitations." https://www.chromium.org/quic
	the first type of time period includes an idle time period that does not include a user timeout period;	Google infringes claim 159, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein the first type of time period (e.g., idle-type timeout, etc.) includes an idle time period that does not include a user timeout period; See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle_timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
		+--------+--------------------------------------+---------------+ | Value | Parameter Name | Specification | +--------+--------------------------------------+---------------+ | 0x0000 | original_connection_id | Section 18.1 | | 0x0001 | idle_timeout | Section 18.1 | | 0x0002 | stateless_reset_token | Section 18.1 | | 0x0003 | max_packet_size | Section 18.1 | | 0x0004 | initial_max_data | Section 18.1 | | 0x0005 | initial_max_stream_data_bidi_local | Section 18.1 | | 0x0006 | initial_max_stream_data_bidi_remote | Section 18.1 | | 0x0007 | initial_max_stream_data_uni | Section 18.1 | | 0x0008 | initial_max_streams_bidi | Section 18.1 | | 0x0009 | initial_max_streams_uni | Section 18.1 | | 0x000a | ack_delay_exponent | Section 18.1 | | 0x000b | max_ack_delay | Section 18.1 | | 0x000c | disable_migration | Section 18.1 | | 0x000d | preferred_address | Section 18.1 | | 0x000e | active_connection_id_limit | Section 18.1 | +--------+--------------------------------------+---------------+ Table 6: Initial QUIC Transport Parameters Entries *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "18. Transport Parameter Encoding

Claim	Claim Elements	Applicability
		The format of the transport parameters is the TransportParameters struct from Figure 15. This is described using the presentation language from Section 3 of [TLS13]. enum { original_connection_id(0), idle_timeout(1), stateless_reset_token(2), max_packet_size(3), initial_max_data(4), initial_max_stream_data_bidi_local(5), initial_max_stream_data_bidi_remote(6), initial_max_stream_data_uni(7), initial_max_streams_bidi(8), initial_max_streams_uni(9), ack_delay_exponent(10), max_ack_delay(11), disable_migration(12), preferred_address(13), active_connection_id_limit(14), (65535) } TransportParameterId;" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily modified based on the value received in the idle_timeout field of the connection negotiation packet.

Claim	Claim Elements	Applicability
		"idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	the first time period is only capable of being detected during the first TCP-variant connection set up;	Google infringes claim 159, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein the first time period (e.g., idle timeout in connection with 0-RTT packets, etc.) is only capable of being detected during the first TCP-variant connection (e.g., QUIC connection, etc.) set up; See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling. 7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake.

Claim	Claim Elements	Applicability
		The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and active_connection_id_limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes

Claim	Claim Elements	Applicability
		initial max data and either initial max streams bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	the first duration is not negotiated between the first and second nodes for the first TCP-variant connection using (a-1), (a-2), nor (a-3), and the second duration and the third duration are negotiated between the first and second nodes for the first TCP-variant connection using at least	Google infringes claim 159, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.) is not negotiated between the first and second nodes for the first TCP-variant connection (e.g., QUIC connection, etc.) using (a-1), (a-2), nor (a-3), and the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.) and the third duration (e.g., duration associated with probe timeout, etc.) are negotiated between the first and second nodes for the first TCP-variant connection using at least one of (a-1), (a-2), or (a-3); See excerpt(s) above and below, for example (emphasis added, if any):

Claim	Claim Elements	Applicability
	one of (a-1), (a-2), or (a-3);	**Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling. 7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and active_connection_id_limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-

Claim	Claim Elements	Applicability
		protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial max data and either initial max streams bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if

Claim	Claim Elements	Applicability
		those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout

Claim	Claim Elements	Applicability
		A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion.

Claim	Claim Elements	Applicability
		A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "QUIC provides reliable, ordered delivery of the cryptographic handshake data. QUIC packet protection is used to encrypt as much of the handshake protocol as possible. The cryptographic handshake MUST provide the following properties: • authenticated key exchange, where ○ a server is always authenticated, ○ a client is optionally authenticated, ○ every connection produces distinct and unrelated keys, ○ keying material is usable for packet protection for both 0-RTT and 1-RTT packets, and ○ 1-RTT keys have forward secrecy • authenticated values for the transport parameters of the peer (see Section 7.3) • authenticated negotiation of an application protocol (TLS uses ALPN [RFC7301] for this purpose)" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	the first duration is exclusively used for detecting the first type of time period until being changed to the second duration;	Google infringes claim 159, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.) is exclusively used for detecting the first type of time period (e.g., idle-type timeout, etc.) until being changed to the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.);

Claim	Claim Elements	Applicability
		See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily detected based on the value received in the idle_timeout field of the connection negotiation packet. "idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10. Connection Termination An established QUIC connection can be terminated in one of three ways: • idle timeout (Section 10.2) • immediate close (Section 10.3) • stateless reset (Section 10.4) An endpoint MAY discard connection state if it does not have a validated path on which it can send packets (see Section 8.2)." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO).

Claim	Claim Elements	Applicability
		Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process.

Claim	Claim Elements	Applicability
		A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and active_connection_id_limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial_max_data and either initial_max_streams_bidi and initial_max_stream_data_bidi_remote, or initial_max_streams_uni and initial_max_stream_data_uni.

Claim	Claim Elements	Applicability
		A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	the second duration is determined based on a first algorithm, and the third duration is determined based on a second algorithm that is different from the first algorithm;	Google infringes claim 159, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.) is determined based on a first algorithm (e.g., used to negotiate an idle timeout period between nodes, etc.), and the third duration (e.g., duration associated with probe timeout, etc.) is determined based on a second algorithm (e.g., used to determine the probe timeout, etc.) that is different from the first algorithm; See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints

Claim	Claim Elements	Applicability
		are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows:

Claim	Claim Elements	Applicability
		PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO).

Claim	Claim Elements	Applicability
		Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 **Note**: Based upon information and belief, keep-alive signals are not required as a result of using idle timers and reducing the number of keep-alive signals to zero and, without a negotiated idle timeout, neither endpoint of a connection knows when the other side might timeout, so keep-alives (e.g., PINGs) have to be sent at unnecessarily shorter intervals to insure the connection to stay active. Further, with a negotiated idle time period, only one side sends one or more keep-alives, if a keep-alive is used. The QUIC IETF transport spec recognizes the need to avoid unnecessary keep-alives (e.g., PINGs). See section 10.1.2 of the QUIC transport draft at https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-34#section-10.1.2 and section 5.1 of the IETF HTTP/3 draft at https://tools.ietf.org/id/draft-ietf-quic-http-34.html#name-idle-connections. "19.2. PING Frame

Claim	Claim Elements	Applicability
		Endpoints can use PING frames (type=0x01) to verify that their peers are still alive or to check reachability to the peer. The PING frame contains no additional fields. The receiver of a PING frame simply needs to acknowledge the packet containing this frame. The PING frame can be used to keep a connection alive when an application or application protocol wishes to prevent the connection from timing out. An application protocol SHOULD provide guidance about the conditions under which generating a PING is recommended. This guidance SHOULD indicate whether it is the client or the server that is expected to send the PING. **Having both endpoints send PING frames without coordination can produce an excessive number of packets and poor performance.** **A connection will time out if no packets are sent or received for a period longer than the time negotiated using the max_idle_timeout transport parameter** (see Section 10). However, state in middleboxes might time out earlier than that. Though REQ-5 in [RFC4787] recommends a 2 minute timeout interval, experience shows that sending packets every 15 to 30 seconds is necessary to prevent the majority of middleboxes from losing state for UDP flows." https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-27#section-19.2
	the third time period is detected based on first TCP-variant connection traffic-related criteria that only involves no acknowledgement packet being received in the first TCP-variant connection during the third duration and processed as an	Google infringes claim 159, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein the third time period (e.g., probe timeout, etc.) is detected based on first TCP-variant connection (e.g., QUIC connection, etc.) traffic-related criteria that only involves no acknowledgement packet being received in the first TCP-variant connection during the third duration (e.g., duration associated with probe timeout, etc.) and processed as an acknowledgement for a sent packet sent in the first TCP-variant connection; See excerpt(s) above and below, for example (emphasis added, if any):

Claim	Claim Elements	Applicability
	acknowledgement for a sent packet sent in the first TCP-variant connection;	"18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3.

Claim	Claim Elements	Applicability
		The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity.

Claim	Claim Elements	Applicability
		The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	during the at least portion of the first TCP-variant connection including the at least portion of the first TCP-variant connection set up, the detection of the first time period of the first type of time period is based on the first duration, by being based on first TCP-variant connection traffic-related criteria that only involves whether any TCP-variant packet is received for the first TCP-variant connection set up during the first duration;	Google infringes claim 159, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein, during the at least portion of the first TCP-variant connection (e.g., QUIC connection, etc.) including the at least portion of the first TCP-variant connection set up, the detection of the first time period (e.g., idle timeout in connection with 0-RTT packets, etc.) of the first type of time period (e.g., idle-type timeout, etc.) is based on the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.), by being based on first TCP-variant connection traffic-related criteria that only involves whether any TCP-variant packet is received for the first TCP-variant connection set up during the first duration; See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints

Claim	Claim Elements	Applicability
		are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original connection id, preferred address, stateless reset token, ack delay exponent and active connection id limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that

Claim	Claim Elements	Applicability
		accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial max data and either initial max streams bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
	based on the first duration being changed to the second duration, the detection of the second time period of the first type of time period is based on the second duration, by being based on first TCP-variant connection traffic-related criteria that only involves whether, during the second duration, no packet is received and processed to keep the first TCP-variant connection at least partially active, nor is expected to be received in the first TCP-variant connection; and	Google infringes claim 159, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein, based on the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.) being changed to the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.), the detection of the second time period (e.g., idle timeout in connection with 1-RTT packets, etc.) of the first type of time period (e.g., idle-type timeout, etc.) is based on the second duration, by being based on first TCP-variant connection (e.g., QUIC connection, etc.) traffic-related criteria that only involves whether, during the second duration, no packet is received (e.g., connection remains idle, etc.) and processed to keep the first TCP-variant connection at least partially active, nor is expected to be received in the first TCP-variant connection; and See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily detected based on the value received in the idle_timeout field of the connection negotiation packet. "idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints

Claim	Claim Elements	Applicability
		are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling. 7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and active_connection_id_limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters.

Claim	Claim Elements	Applicability
		• initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial max data and either initial max streams bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10. Connection Termination

Claim	Claim Elements	Applicability
		An established QUIC connection can be terminated in one of three ways: • idle timeout (Section 10.2) • immediate close (Section 10.3) • stateless reset (Section 10.4) An endpoint MAY discard connection state if it does not have a validated path on which it can send packets (see Section 8.2)." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
		"QUIC provides reliable, ordered delivery of the cryptographic handshake data. QUIC packet protection is used to encrypt as much of the handshake protocol as possible. The cryptographic handshake MUST provide the following properties: • authenticated key exchange, where ○ a server is always authenticated, ○ a client is optionally authenticated, ○ every connection produces distinct and unrelated keys, ○ keying material is usable for packet protection for both 0-RTT and 1-RTT packets, and ○ 1-RTT keys have forward secrecy • authenticated values for the transport parameters of the peer (see Section 7.3) • authenticated negotiation of an application protocol (TLS uses ALPN [RFC7301] for this purpose)" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	an idle timeout attribute is utilized for the first type of time period, and a user timeout attribute, that is separate from the idle timeout attribute, is utilized for controlling communication of one or more acknowledgment packets.	Google infringes claim 159, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein an idle timeout attribute is utilized for the first type of time period (e.g., idle-type timeout, etc.), and a user timeout attribute (e.g., acknowledgement in the form of PING frames, etc.), that is separate from the idle timeout attribute, is utilized for controlling communication of one or more acknowledgment packets. See excerpt(s) above and below, for example (emphasis added, if any): "19.2. PING Frame

Claim	Claim Elements	Applicability
		Endpoints can use PING frames (type=0x01) to verify that their peers are still alive or to check reachability to the peer. The PING frame contains no additional fields. The receiver of a PING frame simply needs to acknowledge the packet containing this frame. The PING frame can be used to keep a connection alive when an application or application protocol wishes to prevent the connection from timing out. An application protocol SHOULD provide guidance about the conditions under which generating a PING is recommended. This guidance SHOULD indicate whether it is the client or the server that is expected to send the PING. Having both endpoints send PING frames without coordination can produce an excessive number of packets and poor performance. A connection will time out if no packets are sent or received for a period longer than the time negotiated using the max_idle_timeout transport parameter (see Section 10). However, state in middleboxes might time out earlier than that. Though REQ-5 in [RFC4787] recommends a 2 minute timeout interval, experience shows that sending packets every 15 to 30 seconds is necessary to prevent the majority of middleboxes from losing state for UDP flows." https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-27#section-19.2
168	The method of claim 167, wherein the third time period is detected independent of the detection of the first and second time periods.	Google infringes claim 167, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein the third time period (e.g., probe timeout, etc.) is detected independent of the detection of the first and second time periods. See excerpt(s) above and below, for example (emphasis added, if any): "18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms

Claim	Claim Elements	Applicability
		firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement.

Claim	Claim Elements	Applicability
		The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a

Claim	Claim Elements	Applicability
		Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
169	The method of claim 167, wherein the second duration is the same as the first duration, despite being capable of being different from the first duration.	Google infringes claim 167, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.) is the same as the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.), despite being capable of being different from the first duration. See excerpt(s) above and below, for example (emphasis added, if any): "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout **can** be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded

Claim	Claim Elements	Applicability
		if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and active_connection_id_limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data.

Claim	Claim Elements	Applicability
		If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial max data and either initial max streams bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL VIOLATION.”

Claim	Claim Elements	Applicability
		QUIC: A UDP-Based Multiplexed and Secure Transport https://tools.ietf.org/html/draft-ietf-quic-transport-22
170	The method of claim 78, wherein: the first TCP-variant packet is not a synchronize (SYN) packet;	Google infringes claim 78, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein: the first TCP-variant packet is not a synchronize (SYN) packet; See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle_timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
		+--------+-----------------------------------+---------------+ | Value | Parameter Name | Specification | +--------+-----------------------------------+---------------+ | 0x0000 | original_connection_id | Section 18.1 | | 0x0001 | idle_timeout | Section 18.1 | | 0x0002 | stateless_reset_token | Section 18.1 | | 0x0003 | max_packet_size | Section 18.1 | | 0x0004 | initial_max_data | Section 18.1 | | 0x0005 | initial_max_stream_data_bidi_local | Section 18.1 | | 0x0006 | initial_max_stream_data_bidi_remote | Section 18.1 | | 0x0007 | initial_max_stream_data_uni | Section 18.1 | | 0x0008 | initial_max_streams_bidi | Section 18.1 | | 0x0009 | initial_max_streams_uni | Section 18.1 | | 0x000a | ack_delay_exponent | Section 18.1 | | 0x000b | max_ack_delay | Section 18.1 | | 0x000c | disable_migration | Section 18.1 | | 0x000d | preferred_address | Section 18.1 | | 0x000e | active_connection_id_limit | Section 18.1 | +--------+-----------------------------------+---------------+ Table 6: Initial QUIC Transport Parameters Entries *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 **Note**: The following serves to distinguish the use of SYN handshake packets from QUIC negotiation packets where parameter values like idle_timeout may be exchanged.

Claim	Claim Elements	Applicability
		"The synchronization requires each side to send its own initial sequence number and to receive a confirmation of it in acknowledgment from the other side. Each side must also receive the other side's initial sequence number and send a confirming acknowledgment. 1) A --> B SYN my sequence number is X 2) A <-- B ACK your sequence number is X 3) A <-- B SYN my sequence number is Y 4) A --> B ACK your sequence number is Y Because steps 2 and 3 can be combined in a single message this is called the three way (or three message) handshake. A three way handshake is necessary because sequence numbers are not tied to a global clock in the network, and TCPs may have different mechanisms for picking the ISN's. The receiver of the first SYN has no way of knowing whether the segment was an old delayed one or not, unless it remembers the last sequence number used on the connection (which is not always possible), and so it must ask the sender to verify this SYN. The three way handshake and the advantages of a clock-driven scheme are discussed in [3]." "Request for Comments" (RFC) document RFC 793 (September 1981) https://tools.ietf.org/html/rfc793
	the first TCP-variant connection is not a TCP extension connection that involves a TCP extension;	Google infringes claim 78, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein the first TCP-variant connection (e.g., QUIC connection, etc.) is not a TCP extension connection that involves a TCP extension; See excerpt(s) above and below, for example (emphasis added, if any): **Note**: The following evidence is related to and/or describes the QUIC standard contained herein.

Claim	Claim Elements	Applicability
		"On the surface, QUIC is very similar to TCP+TLS+HTTP/2 implemented on UDP. ...However, since QUIC is built on top of UDP, it suffers from no such limitations." https://www.chromium.org/quic
	the first type of time period includes an idle time period that does not include a user timeout period;	Google infringes claim 78, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein the first type of time period (e.g., idle-type timeout, etc.) includes an idle time period that does not include a user timeout period; See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle_timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability

Value	Parameter Name	Specification
0x0000	original_connection_id	Section 18.1
0x0001	idle_timeout	Section 18.1
0x0002	stateless_reset_token	Section 18.1
0x0003	max_packet_size	Section 18.1
0x0004	initial_max_data	Section 18.1
0x0005	initial_max_stream_data_bidi_local	Section 18.1
0x0006	initial_max_stream_data_bidi_remote	Section 18.1
0x0007	initial_max_stream_data_uni	Section 18.1
0x0008	initial_max_streams_bidi	Section 18.1
0x0009	initial_max_streams_uni	Section 18.1
0x000a	ack_delay_exponent	Section 18.1
0x000b	max_ack_delay	Section 18.1
0x000c	disable_migration	Section 18.1
0x000d	preferred_address	Section 18.1
0x000e	active_connection_id_limit	Section 18.1

Table 6: Initial QUIC Transport Parameters Entries

QUIC: A UDP-Based Multiplexed and Secure Transport https://tools.ietf.org/html/draft-ietf-quic-transport-22

"18. Transport Parameter Encoding

Claim	Claim Elements	Applicability
		The format of the transport parameters is the TransportParameters struct from Figure 15. This is described using the presentation language from Section 3 of [TLS13]. enum { original_connection_id(0), idle_timeout(1), stateless_reset_token(2), max_packet_size(3), initial_max_data(4), initial_max_stream_data_bidi_local(5), initial_max_stream_data_bidi_remote(6), initial_max_stream_data_uni(7), initial_max_streams_bidi(8), initial_max_streams_uni(9), ack_delay_exponent(10), max_ack_delay(11), disable_migration(12), preferred_address(13), active_connection_id_limit(14), (65535) } TransportParameterId;" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily modified based on the value received in the idle_timeout field of the connection negotiation packet.

Claim	Claim Elements	Applicability
		"idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	the first time period is only capable of being detected during the first TCP-variant connection set up;	Google infringes claim 78, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein the first time period (e.g., idle timeout in connection with 0-RTT packets, etc.) is only capable of being detected during the first TCP-variant connection (e.g., QUIC connection, etc.) set up; See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling. 7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake.

Claim	Claim Elements	Applicability
		The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and active_connection_id_limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes

Claim	Claim Elements	Applicability
		initial max data and either initial max streams bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	the first duration is not negotiated between the first and second nodes for the first TCP-variant connection using (a-1), (a-2), nor (a-3), and the second duration and the third duration are negotiated between the first and second nodes for the first TCP-variant connection using at least	Google infringes claim 78, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.) is not negotiated between the first and second nodes for the first TCP-variant connection (e.g., QUIC connection, etc.) using (a-1), (a-2), nor (a-3), and the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.) and the third duration (e.g., duration associated with probe timeout, etc.) are negotiated between the first and second nodes for the first TCP-variant connection using at least one of (a-1), (a-2), or (a-3); See excerpt(s) above and below, for example (emphasis added, if any):

Claim	Claim Elements	Applicability
	one of (a-1), (a-2), or (a-3);	**Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling. 7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and active_connection_id_limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-

Claim	Claim Elements	Applicability
		protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial max data and either initial max streams bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if

Claim	Claim Elements	Applicability
		those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout

Claim	Claim Elements	Applicability
		A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion.

Claim	Claim Elements	Applicability
		A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "QUIC provides reliable, ordered delivery of the cryptographic handshake data. QUIC packet protection is used to encrypt as much of the handshake protocol as possible. The cryptographic handshake MUST provide the following properties: • authenticated key exchange, where ○ a server is always authenticated, ○ a client is optionally authenticated, ○ every connection produces distinct and unrelated keys, ○ keying material is usable for packet protection for both 0-RTT and 1-RTT packets, and ○ 1-RTT keys have forward secrecy • authenticated values for the transport parameters of the peer (see Section 7.3) • authenticated negotiation of an application protocol (TLS uses ALPN [RFC7301] for this purpose)" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	the first duration is exclusively used for detecting the first type of time period until being changed to the second duration;	Google infringes claim 78, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.) is exclusively used for detecting the first type of time period (e.g., idle-type timeout, etc.) until being changed to the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.);

Claim	Claim Elements	Applicability
		See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily detected based on the value received in the idle_timeout field of the connection negotiation packet. "idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10. Connection Termination An established QUIC connection can be terminated in one of three ways: • idle timeout (Section 10.2) • immediate close (Section 10.3) • stateless reset (Section 10.4) An endpoint MAY discard connection state if it does not have a validated path on which it can send packets (see Section 8.2)." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO).

Claim	Claim Elements	Applicability
		Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process.

Claim	Claim Elements	Applicability
		A client MUST NOT use remembered values for the following parameters: original connection id, preferred address, stateless reset token, ack delay exponent and active connection id limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial max data and either initial max streams bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni.

Claim	Claim Elements	Applicability
		A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	the second duration is determined based on a first algorithm, and the third duration is determined based on a second algorithm that is different from the first algorithm;	Google infringes claim 78, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.) is determined based on a first algorithm (e.g., used to negotiate an idle timeout period between nodes, etc.), and the third duration (e.g., duration associated with probe timeout, etc.) is determined based on a second algorithm (e.g., used to determine the probe timeout, etc.) that is different from the first algorithm; See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints

Claim	Claim Elements	Applicability
		are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows:

Claim	Claim Elements	Applicability
		PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO).

Claim	Claim Elements	Applicability
		Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 **Note**: Based upon information and belief, keep-alive signals are not required as a result of using idle timers and reducing the number of keep-alive signals to zero and, without a negotiated idle timeout, neither endpoint of a connection knows when the other side might timeout, so keep-alives (e.g., PINGs) have to be sent at unnecessarily shorter intervals to insure the connection to stay active. Further, with a negotiated idle time period, only one side sends one or more keep-alives, if a keep-alive is used. The QUIC IETF transport spec recognizes the need to avoid unnecessary keep-alives (e.g., PINGs). See section 10.1.2 of the QUIC transport draft at https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-34#section-10.1.2 and section 5.1 of the IETF HTTP/3 draft at https://tools.ietf.org/id/draft-ietf-quic-http-34.html#name-idle-connections. "19.2. PING Frame

Claim	Claim Elements	Applicability
		Endpoints can use PING frames (type=0x01) to verify that their peers are still alive or to check reachability to the peer. The PING frame contains no additional fields. The receiver of a PING frame simply needs to acknowledge the packet containing this frame. The PING frame can be used to keep a connection alive when an application or application protocol wishes to prevent the connection from timing out. An application protocol SHOULD provide guidance about the conditions under which generating a PING is recommended. This guidance SHOULD indicate whether it is the client or the server that is expected to send the PING. **Having both endpoints send PING frames without coordination can produce an excessive number of packets and poor performance.** **A connection will time out if no packets are sent or received for a period longer than the time negotiated using the max_idle_timeout transport parameter** (see Section 10). However, state in middleboxes might time out earlier than that. Though REQ-5 in [RFC4787] recommends a 2 minute timeout interval, experience shows that sending packets every 15 to 30 seconds is necessary to prevent the majority of middleboxes from losing state for UDP flows." https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-27#section-19.2
	the third time period is detected based on first TCP-variant connection traffic-related criteria that only involves, during the third duration, no acknowledgement packet being received in the first TCP-variant connection and processed as an	Google infringes claim 78, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein the third time period (e.g., probe timeout, etc.) is detected based on first TCP-variant connection (e.g., QUIC connection, etc.) traffic-related criteria that only involves, during the third duration (e.g., duration associated with probe timeout, etc.), no acknowledgement packet being received in the first TCP-variant connection and processed as an acknowledgement for a sent packet sent in the first TCP-variant connection; See excerpt(s) above and below, for example (emphasis added, if any):

Claim	Claim Elements	Applicability
	acknowledgement for a sent packet sent in the first TCP-variant connection;	"18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3.

Claim	Claim Elements	Applicability
		The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity.

Claim	Claim Elements	Applicability
		The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	during the at least portion of the first TCP-variant connection including the at least portion of the first TCP-variant connection set up, the detection of the first time period of the first type of time period is based on the first duration, by being based on first TCP-variant connection traffic-related criteria that only involves whether any TCP-variant packet is received or sent for the first TCP-variant connection set up during the first duration;	Google infringes claim 78, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein, during the at least portion of the first TCP-variant connection (e.g., QUIC connection, etc.) including the at least portion of the first TCP-variant connection set up, the detection of the first time period (e.g., idle timeout in connection with 0-RTT packets, etc.) of the first type of time period (e.g., idle-type timeout, etc.) is based on the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.), by being based on first TCP-variant connection traffic-related criteria that only involves whether any TCP-variant packet is received or sent for the first TCP-variant connection set up during the first duration; See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints

Claim	Claim Elements	Applicability
		are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and active_connection_id_limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that

Claim	Claim Elements	Applicability
		accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial max data and either initial max streams bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
	based on the first duration being changed to the second duration, the detection of the second time period of the first type of time period is based on the second duration, by being based on first TCP-variant connection traffic-related criteria that only involves whether, during the second duration, no packet is received and processed to keep the first TCP-variant connection at least partially active, sent, nor is expected to be received or sent in the first TCP-variant connection; and	Google infringes claim 78, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein, based on the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.) being changed to the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.), the detection of the second time period (e.g., idle timeout in connection with 1-RTT packets, etc.) of the first type of time period (e.g., idle-type timeout, etc.) is based on the second duration, by being based on first TCP-variant connection (e.g., QUIC connection, etc.) traffic-related criteria that only involves whether, during the second duration, no packet is received (e.g., connection remains idle, etc.) and processed to keep the first TCP-variant connection at least partially active, sent, nor is expected to be received or sent in the first TCP-variant connection; and See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily detected based on the value received in the idle_timeout field of the connection negotiation packet. "idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints

Claim	Claim Elements	Applicability
		are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling. 7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and active_connection_id_limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters.

Claim	Claim Elements	Applicability
		• initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial max data and either initial max streams bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10. Connection Termination

Claim	Claim Elements	Applicability
		An established QUIC connection can be terminated in one of three ways: • idle timeout (Section 10.2) • immediate close (Section 10.3) • stateless reset (Section 10.4) An endpoint MAY discard connection state if it does not have a validated path on which it can send packets (see Section 8.2)." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
		"QUIC provides reliable, ordered delivery of the cryptographic handshake data. QUIC packet protection is used to encrypt as much of the handshake protocol as possible. The cryptographic handshake MUST provide the following properties: • authenticated key exchange, where ○ a server is always authenticated, ○ a client is optionally authenticated, ○ every connection produces distinct and unrelated keys, ○ keying material is usable for packet protection for both 0-RTT and 1-RTT packets, and ○ 1-RTT keys have forward secrecy • authenticated values for the transport parameters of the peer (see Section 7.3) • authenticated negotiation of an application protocol (TLS uses ALPN [RFC7301] for this purpose)" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	an idle timeout attribute is utilized for the first type of time period, and a user timeout attribute, that is separate from the idle timeout attribute, is utilized for controlling communication of one or more acknowledgment packets.	Google infringes claim 78, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein an idle timeout attribute is utilized for the first type of time period (e.g., idle-type timeout, etc.), and a user timeout attribute (e.g., acknowledgement in the form of PING frames, etc.), that is separate from the idle timeout attribute, is utilized for controlling communication of one or more acknowledgment packets. See excerpt(s) above and below, for example (emphasis added, if any): "19.2. PING Frame

Claim	Claim Elements	Applicability
		Endpoints can use PING frames (type=0x01) to verify that their peers are still alive or to check reachability to the peer. The PING frame contains no additional fields. The receiver of a PING frame simply needs to acknowledge the packet containing this frame. The PING frame can be used to keep a connection alive when an application or application protocol wishes to prevent the connection from timing out. An application protocol SHOULD provide guidance about the conditions under which generating a PING is recommended. This guidance SHOULD indicate whether it is the client or the server that is expected to send the PING. Having both endpoints send PING frames without coordination can produce an excessive number of packets and poor performance. A connection will time out if no packets are sent or received for a period longer than the time negotiated using the max_idle_timeout transport parameter (see Section 10). However, state in middleboxes might time out earlier than that. Though REQ-5 in [RFC4787] recommends a 2 minute timeout interval, experience shows that sending packets every 15 to 30 seconds is necessary to prevent the majority of middleboxes from losing state for UDP flows." https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-27#section-19.2
171	The method of claim 78, wherein the third time period is detected based on the third duration, by being detected when no acknowledgement packet is detected as being received and processed, during the third duration, by the first node in the first TCP-variant connection to	Google infringes claim 78, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein the third time period (e.g., probe timeout, etc.) is detected based on the third duration (e.g., duration associated with probe timeout, etc.), by being detected when no acknowledgement packet is detected as being received and processed, during the third duration, by the first node (e.g., one of a QUIC-compliant server or client, etc.) in the first TCP-variant connection (e.g., QUIC connection, etc.) to acknowledge a sent packet of a plurality of sent packets sent by the first node, where the third duration starts for each of the plurality of sent packets when at least one of the plurality of sent packets is sent, and where a single timer and a single timeout variable are utilized for detecting the third time period in connection with the plurality of sent packets.

Claim	Claim Elements	Applicability
	acknowledge a sent packet of a plurality of sent packets sent by the first node, where the third duration starts for each of the plurality of sent packets when at least one of the plurality of sent packets is sent, and where a single timer and a single timeout variable are utilized for detecting the third time period in connection with the plurality of sent packets.	See excerpt(s) above and below, for example (emphasis added, if any): "18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay

Claim	Claim Elements	Applicability
		kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an

Claim	Claim Elements	Applicability
		ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
172	The method of claim 171, wherein the third duration starts for each of the plurality of sent packets when the each one of the plurality of sent packets is sent.	Google infringes claim 171, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein the third duration (e.g., duration associated with probe timeout, etc.) starts for each of the plurality of sent packets when the each one of the plurality of sent packets is sent. See excerpt(s) above and below, for example (emphasis added, if any): "18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid"

Claim	Claim Elements	Applicability
		QUIC: A UDP-Based Multiplexed and Secure Transport https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately.

Claim	Claim Elements	Applicability
		When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried."

Claim	Claim Elements	Applicability
		QUIC: A UDP-Based Multiplexed and Secure Transport https://tools.ietf.org/html/draft-ietf-quic-transport-22
173	The method of claim 171, wherein the third duration starts for each of the plurality of sent packets when a first one of the plurality of sent packets is sent.	Google infringes claim 171, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein the third duration (e.g., duration associated with probe timeout, etc.) starts for each of the plurality of sent packets when a first one of the plurality of sent packets is sent. See excerpt(s) above and below, for example (emphasis added, if any): "18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298].

Claim	Claim Elements	Applicability
		6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "10.2. Idle Timeout

Claim	Claim Elements	Applicability
		If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
174	The method of claim 171, wherein the third duration starts for each of the plurality of sent packets when a last one of the plurality of sent packets is sent or received.	Google infringes claim 171, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein the third duration (e.g., duration associated with probe timeout, etc.) starts for each of the plurality of sent packets when a last one of the plurality of sent packets is sent or received. See excerpt(s) above and below, for example (emphasis added, if any): "18.1 Transport Parameter Definitions …

Claim	Claim Elements	Applicability
		• max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt),

Claim	Claim Elements	Applicability
		the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity.

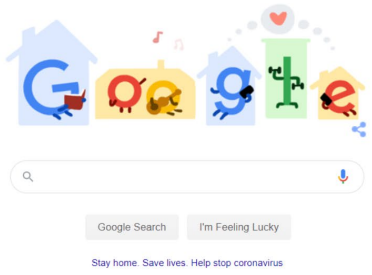
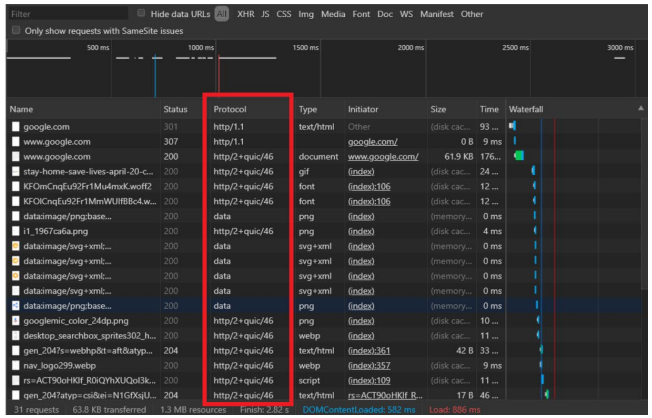
Claim	Claim Elements	Applicability
		The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
175	The method of claim 171, wherein the third duration starts for each of the plurality of sent packets when a first one of the plurality of sent packets is sent, where the first packet is a first packet to be received after a last one of the plurality of sent packets is sent.	Google infringes claim 171, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform to perform a method wherein the third duration (e.g., duration associated with probe timeout, etc.) starts for each of the plurality of sent packets when a first one of the plurality of sent packets is sent, where the first packet is a first packet to be received after a last one of the plurality of sent packets is sent. See excerpt(s) above and below, for example (emphasis added, if any): "18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to 1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
		"6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement. The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion.

Claim	Claim Elements	Applicability
		A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
176	An apparatus, comprising: a non-transitory memory means for storing instructions means; and one or more processors means in communication with the non-transitory memory means, wherein the one or more processors means execute the instructions means to:	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) comprising: a non-transitory memory means for storing instructions (e.g., server software, etc.) means; and one or more processors means in communication with the non-transitory memory means, wherein the one or more processors means execute the instructions means to: See excerpt(s) above and below, for example (emphasis added, if any): **Note**: Google admits that their "Google Cloud Platform" uses QUIC: "Google Cloud Platform Blog... we're happy to be the first major public cloud to offer QUIC support for our HTTPS load balancers." https://cloudplatform.googleblog.com/2018/06/Introducing-QUIC-support-for-HTTPS-load-balancing.html **Note**: The "Google Cloud Platform" that, as established above, uses QUIC, is also used for ALL of Google's services: "Google Cloud Platform (GCP), offered by Google (company), is a suite of cloud computing services that runs on the same infrastructure that Google uses internally for its end-user products, such as Google Search, Gmail, file storage, and YouTube" https://en.wikipedia.org/wiki/Google_Cloud_Platform **Note**: At least a portion of the citations herein are made to the QUIC standard found at: https://tools.ietf.org/html/draft-ietf-quic-transport-22, and/or https://tools.ietf.org/html/draft-ietf-quic-transport-27. Based upon information and belief, the pertinent portions of such version of the QUIC standard may also be found in one or more other versions. For completeness, all of the versions below (and future iterations thereof) are hereby incorporated by reference:

Claim	Claim Elements	Applicability
		https://tools.ietf.org/html/draft-ietf-quic-transport-00 https://tools.ietf.org/html/draft-ietf-quic-transport-01 https://tools.ietf.org/html/draft-ietf-quic-transport-02 https://tools.ietf.org/html/draft-ietf-quic-transport-03 https://tools.ietf.org/html/draft-ietf-quic-transport-04 https://tools.ietf.org/html/draft-ietf-quic-transport-05 https://tools.ietf.org/html/draft-ietf-quic-transport-06 https://tools.ietf.org/html/draft-ietf-quic-transport-07 https://tools.ietf.org/html/draft-ietf-quic-transport-08 https://tools.ietf.org/html/draft-ietf-quic-transport-09 https://tools.ietf.org/html/draft-ietf-quic-transport-10 https://tools.ietf.org/html/draft-ietf-quic-transport-11 https://tools.ietf.org/html/draft-ietf-quic-transport-12 https://tools.ietf.org/html/draft-ietf-quic-transport-13 https://tools.ietf.org/html/draft-ietf-quic-transport-14 https://tools.ietf.org/html/draft-ietf-quic-transport-15 https://tools.ietf.org/html/draft-ietf-quic-transport-16 https://tools.ietf.org/html/draft-ietf-quic-transport-17 https://tools.ietf.org/html/draft-ietf-quic-transport-18 https://tools.ietf.org/html/draft-ietf-quic-transport-19 https://tools.ietf.org/html/draft-ietf-quic-transport-20 https://tools.ietf.org/html/draft-ietf-quic-transport-21 https://tools.ietf.org/html/draft-ietf-quic-transport-22 https://tools.ietf.org/html/draft-ietf-quic-transport-23 https://tools.ietf.org/html/draft-ietf-quic-transport-24 https://tools.ietf.org/html/draft-ietf-quic-transport-25 https://tools.ietf.org/html/draft-ietf-quic-transport-26 https://tools.ietf.org/html/draft-ietf-quic-transport-27

Claim	Claim Elements	Applicability
		Note: Though these contentions may refer to a specific version of the QUIC standard, the presented evidence is not exclusive to this specific version and is compliant with and/or supported by current versions of the QUIC standard, including QUIC_VERSION_IETF_RFC_V1 (https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-34), QUIC_VERSION_IETF_DRAFT_25 (https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-25), QUIC_VERSION_IETF_DRAFT_27 (https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-27), QUIC_VERSION_IETF_DRAFT_28 (https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-28), QUIC_VERSION_IETF_DRAFT_29 (https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-29), and/or QUIC_VERSION_50 which relies in part on IETF_QUIC_INVARIANTS_06 (https://tools.ietf.org/html/draft-ietf-quic-invariants-06) and/or IETF_QUIC_TRANSPORT_17 (https://tools.ietf.org/html/draft-ietf-quic-transport-17), each of which, upon information and belief, has been utilized by Google LLC through its accused instrumentalities. **Note**: Below is a web page of Google (https://www.google.com/).  

Claim	Claim Elements	Applicability
	identify, at a first node, first information on which at least a first duration for detecting a first type of time period is based;	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) configured to identify, at a first node (e.g., one of a QUIC-compliant server or client, etc.), first information (e.g., first value, etc.) on which at least a first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.) for detecting a first type of time period (e.g., idle-type timeout, etc.) is based; See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily detected based on the value received in the idle_timeout field of the connection negotiation packet. "idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10. Connection Termination An established QUIC connection can be terminated in one of three ways: • idle timeout (Section 10.2) • immediate close (Section 10.3) • stateless reset (Section 10.4) An endpoint MAY discard connection state if it does not have a validated path on which it can send packets (see Section 8.2)." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout

Claim	Claim Elements	Applicability
		If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake.

Claim	Claim Elements	Applicability
		The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original connection id, preferred address, stateless reset token, ack delay exponent and active connection id limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial_max_data and either initial_max_streams_bidi and

Claim	Claim Elements	Applicability
		initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "3.3. Session resumption versus Keep-alive Because QUIC is encapsulated in UDP, applications using QUIC must deal with short idle timeouts. Deployed stateful middleboxes will generally establish state for UDP flows on the first packet state, and keep state for much shorter idle periods than for TCP. According to a 2010 study ([Hatonen10]), UDP applications can assume that any NAT binding or other state entry will be expired after just thirty seconds of inactivity. A QUIC application has three strategies to deal with this issue: • Ignore it, if the application-layer protocol consists only of interactions with no or very short idle periods. • Ensure there are no long idle periods.

Claim	Claim Elements	Applicability
		• Resume the session after a long idle period, using 0-RTT resumption when appropriate. ... Alternatively, the client (but not the server) can use session resumption instead of sending keepalive traffic. In this case, a client that wants to send data to a server over a connection idle longer than the server's idle timeout (available from the idle timeout transport parameter) can simply reconnect. When possible, this reconnection can use 0-RTT session resumption, reducing the latency involved with restarting the connection. This of course only applies in cases in which 0-RTT data is safe, when the client is the restarting peer, and when the data to be sent is idempotent." https://www.ietf.org/id/draft-ietf-quic-applicability-07.txt
	allocate a first resource for a first non-Transmission Control Protocol (non-TCP) connection;	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) configured to allocate a first resource (e.g. a storage and/or data structure for storing connection IDs, a connection state, send/recv buffers, etc.) for a first non-Transmission Control Protocol (non-TCP) connection (e.g., QUIC connection, etc.); See excerpt(s) above and below, for example (emphasis added, if any): "Google Cloud Platform (GCP), offered by Google (company), is a suite of cloud computing services that runs on the same infrastructure that Google uses internally for its end-user products, such as Google Search, Gmail, file storage, and YouTube" https://en.wikipedia.org/wiki/Google_Cloud_Platform
	generate a first non-TCP packet including a first parameter field identifying first metadata for use in determining a second duration for detecting the first type of time period;	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) configured to generate a first non-TCP packet (e.g., first QUIC negotiation packet, etc.) including a first parameter field (e.g., first idle_timeout parameter field, etc.) identifying first metadata (e.g., first idle timeout value, etc.) for use in determining a second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.) for detecting the first type of time period (e.g., idle-type timeout, etc.);

Claim	Claim Elements	Applicability
		See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability

Value	Parameter Name	Specification
0x0000	original_connection_id	Section 18.1
0x0001	idle_timeout	Section 18.1
0x0002	stateless_reset_token	Section 18.1
0x0003	max_packet_size	Section 18.1
0x0004	initial_max_data	Section 18.1
0x0005	initial_max_stream_data_bidi_local	Section 18.1
0x0006	initial_max_stream_data_bidi_remote	Section 18.1
0x0007	initial_max_stream_data_uni	Section 18.1
0x0008	initial_max_streams_bidi	Section 18.1
0x0009	initial_max_streams_uni	Section 18.1
0x000a	ack_delay_exponent	Section 18.1
0x000b	max_ack_delay	Section 18.1
0x000c	disable_migration	Section 18.1
0x000d	preferred_address	Section 18.1
0x000e	active_connection_id_limit	Section 18.1

Table 6: Initial QUIC Transport Parameters Entries

QUIC: A UDP-Based Multiplexed and Secure Transport https://tools.ietf.org/html/draft-ietf-quic-transport-22

"18. Transport Parameter Encoding

Claim	Claim Elements	Applicability
		The format of the transport parameters is the TransportParameters struct from Figure 15. This is described using the presentation language from Section 3 of [TLS13]. enum { original_connection_id(0), idle_timeout(1), stateless_reset_token(2), max_packet_size(3), initial_max_data(4), initial_max_stream_data_bidi_local(5), initial_max_stream_data_bidi_remote(6), initial_max_stream_data_uni(7), initial_max_streams_bidi(8), initial_max_streams_uni(9), ack_delay_exponent(10), max_ack_delay(11), disable_migration(12), preferred_address(13), active_connection_id_limit(14), (65535) } TransportParameterId;" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily modified based on the value received in the idle_timeout field of the connection negotiation packet.

Claim	Claim Elements	Applicability
		"idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "QUIC provides reliable, ordered delivery of the cryptographic handshake data. QUIC packet protection is used to encrypt as much of the handshake protocol as possible. The cryptographic handshake MUST provide the following properties: • authenticated key exchange, where ○ a server is always authenticated, ○ a client is optionally authenticated, ○ every connection produces distinct and unrelated keys, ○ keying material is usable for packet protection for both 0-RTT and 1-RTT packets, and ○ 1-RTT keys have forward secrecy • authenticated values for the transport parameters of the peer (see Section 7.3) • authenticated negotiation of an application protocol (TLS uses ALPN [RFC7301] for this purpose)" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	set up the first non-TCP connection, by sending, from the first node to a second node, the first non-TCP packet to provide the first metadata to the	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) configured to set up the first non-TCP connection (e.g., QUIC connection, etc.), by sending, from the first node (e.g., one of a QUIC-compliant server or client, etc.) to a second node (e.g., the other one of the QUIC-compliant server or client, etc.), the first non-TCP packet (e.g., first QUIC negotiation packet, etc.) to provide the first metadata (e.g., first idle timeout value, etc.) to the second node, for use by the second node in determining the

Claim	Claim Elements	Applicability
	second node, for use by the second node in determining the second duration for detecting the first type of time period;	second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.) for detecting the first type of time period (e.g., idle-type timeout, etc.); See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, a QUIC negotiation packet includes transport parameters that include an idle_timeout parameter that is detected by a recipient of such packet. "7.3. Transport Parameters During connection establishment, both endpoints make authenticated declarations of their transport parameters. These declarations are made unilaterally by each endpoint. Endpoints are required to comply with the restrictions implied by these parameters; the description of each parameter includes rules for its handling." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability

Value	Parameter Name	Specification
0x0000	original_connection_id	Section 18.1
0x0001	idle_timeout	Section 18.1
0x0002	stateless_reset_token	Section 18.1
0x0003	max_packet_size	Section 18.1
0x0004	initial_max_data	Section 18.1
0x0005	initial_max_stream_data_bidi_local	Section 18.1
0x0006	initial_max_stream_data_bidi_remote	Section 18.1
0x0007	initial_max_stream_data_uni	Section 18.1
0x0008	initial_max_streams_bidi	Section 18.1
0x0009	initial_max_streams_uni	Section 18.1
0x000a	ack_delay_exponent	Section 18.1
0x000b	max_ack_delay	Section 18.1
0x000c	disable_migration	Section 18.1
0x000d	preferred_address	Section 18.1
0x000e	active_connection_id_limit	Section 18.1

Table 6: Initial QUIC Transport Parameters Entries

QUIC: A UDP-Based Multiplexed and Secure Transport https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
		"QUIC provides reliable, ordered delivery of the cryptographic handshake data. QUIC packet protection is used to encrypt as much of the handshake protocol as possible. The cryptographic handshake MUST provide the following properties: • authenticated key exchange, where ○ a server is always authenticated, ○ a client is optionally authenticated, ○ every connection produces distinct and unrelated keys, ○ keying material is usable for packet protection for both 0-RTT and 1-RTT packets, and ○ 1-RTT keys have forward secrecy • authenticated values for the transport parameters of the peer (see Section 7.3) • authenticated negotiation of an application protocol (TLS uses ALPN [RFC7301] for this purpose)" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	in response to detecting, based on the first duration and by the first node during at least a portion of the first non-TCP connection including at least a portion of the first non-TCP connection set up, a first time period of the first type of time period, at least partially close the first non-TCP	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) configured to, in response to detecting, based on the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.) and by the first node (e.g., one of a QUIC-compliant server or client, etc.) during at least a portion of the first non-TCP connection (e.g., QUIC connection, etc.) including at least a portion of the first non-TCP connection set up, a first time period (e.g., idle timeout in connection with 0-RTT packets, etc.) of the first type of time period (e.g., idle-type timeout, etc.), at least partially close (e.g., terminate, etc.) the first non-TCP connection, by releasing, by the first node, the first resource (e.g. a storage and/or data structure for storing connection IDs, a connection state, send/recv buffers, etc.) allocated for the first non-TCP connection; See excerpt(s) above and below, for example (emphasis added, if any):

Claim	Claim Elements	Applicability
	connection, by releasing, by the first node, the first resource allocated for the first non-TCP connection;	**Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily detected based on the value received in the idle_timeout field of the connection negotiation packet. "idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10. Connection Termination An established QUIC connection can be terminated in one of three ways: • idle timeout (Section 10.2) • immediate close (Section 10.3) • stateless reset (Section 10.4) An endpoint MAY discard connection state if it does not have a validated path on which it can send packets (see Section 8.2)." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO)." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
		"7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake. The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and active_connection_id_limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote

Claim	Claim Elements	Applicability
		• initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes initial_max_data and either initial_max_streams_bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	in response to detecting, based on the second duration and by the first node after the first duration is changed to the	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) configured to, in response to detecting, based on the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.) and by the first node (e.g., one of a QUIC-compliant server or client, etc.) after the first duration (e.g., duration associated with idle timeout in connection with 0-RTT packets, etc.) is changed to

Claim	Claim Elements	Applicability
	second duration, a second time period of the first type of time period, at least partially close the first non-TCP connection, by releasing, by the first node, the first resource allocated for the first non-TCP connection;	the second duration, a second time period (e.g., idle timeout in connection with 1-RTT packets, etc.) of the first type of time period (e.g., idle-type timeout, etc.), at least partially close (e.g., terminate, etc.) the first non-TCP connection (e.g., QUIC connection, etc.), by releasing, by the first node, the first resource (e.g. a storage and/or data structure for storing connection IDs, a connection state, send/recv buffers, etc.) allocated for the first non-TCP connection; See excerpt(s) above and below, for example (emphasis added, if any): **Note**: As set forth below, since the idle_timeout value sets the duration of idleness, after which the connection is shutdown, a timeout attribute of the connection is necessarily detected based on the value received in the idle_timeout field of the connection negotiation packet. "idle_timeout (0x0001): The idle timeout is a value in milliseconds that is encoded as an integer; see (Section 10.2). If this parameter is absent or zero then the idle timeout is disabled." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "10. Connection Termination An established QUIC connection can be terminated in one of three ways: • idle timeout (Section 10.2) • immediate close (Section 10.3) • stateless reset (Section 10.4) An endpoint MAY discard connection state if it does not have a validated path on which it can send packets (see Section 8.2)." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22

Claim	Claim Elements	Applicability
		"10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "7.3.1. Values of Transport Parameters for 0-RTT Both endpoints store the value of the server transport parameters from a connection and apply them to any 0-RTT packets that are sent in subsequent connections to that peer, except for transport parameters that are explicitly excluded. Remembered transport parameters apply to the new connection until the handshake completes and the client starts sending 1-RTT packets. Once the handshake completes, the client uses the transport parameters established in the handshake.

Claim	Claim Elements	Applicability
		The definition of new transport parameters (Section 7.3.2) MUST specify whether they MUST, MAY, or MUST NOT be stored for 0-RTT. A client need not store a transport parameter it cannot process. A client MUST NOT use remembered values for the following parameters: original_connection_id, preferred_address, stateless_reset_token, ack_delay_exponent and active_connection_id_limit. The client MUST use the server's new values in the handshake instead, and absent new values from the server, the default value. A client that attempts to send 0-RTT data MUST remember all other transport parameters used by the server. The server can remember these transport parameters, or store an integrity-protected copy of the values in the ticket and recover the information when accepting 0-RTT data. A server uses the transport parameters in determining whether to accept 0-RTT data. If 0-RTT data is accepted by the server, the server MUST NOT reduce any limits or alter any values that might be violated by the client with its 0-RTT data. In particular, a server that accepts 0-RTT data MUST NOT set values for the following parameters (Section 18.1) that are smaller than the remembered value of the parameters. • initial_max_data • initial_max_stream_data_bidi_local • initial_max_stream_data_bidi_remote • initial_max_stream_data_uni • initial_max_streams_bidi • initial_max_streams_uni Omitting or setting a zero value for certain transport parameters can result in 0-RTT data being enabled, but not usable. The applicable subset of transport parameters that permit sending of application data SHOULD be set to non-zero values for 0-RTT. This includes

Claim	Claim Elements	Applicability
		initial max data and either initial max streams bidi and initial max stream data bidi remote, or initial max streams uni and initial max stream data uni. A server MUST either reject 0-RTT data or abort a handshake if the implied values for transport parameters cannot be supported. When sending frames in 0-RTT packets, a client MUST only use remembered transport parameters; importantly, it MUST NOT use updated values that it learns from the server's updated transport parameters or from frames received in 1-RTT packets. Updated values of transport parameters from the handshake apply only to 1-RTT packets. For instance, flow control limits from remembered transport parameters apply to all 0-RTT packets even if those values are increased by the handshake or by frames sent in 1-RTT packets. A server MAY treat use of updated transport parameters in 0-RTT as a connection error of type PROTOCOL_VIOLATION." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22
	in response to detecting, based on a third duration and by the first node, a third time period, at least partially close the first non-TCP connection, by releasing, by the first node, the first resource allocated for the first non-TCP connection, where the third duration is determined based on a first algorithm that is different	Google uses an apparatus (e.g., a platform or device, including a QUIC-compliant server or client, etc.) configured to, in response to detecting, based on a third duration (e.g., duration associated with probe timeout, etc.) and by the first node (e.g., one of a QUIC-compliant server or client, etc.), a third time period (e.g., probe timeout, etc.), at least partially close (e.g., terminate, etc.) the first non-TCP connection (e.g., QUIC connection, etc.), by releasing, by the first node, the first resource (e.g. a storage and/or data structure for storing connection IDs, a connection state, send/recv buffers, etc.) allocated for the first non-TCP connection, where the third duration is determined based on a first algorithm (e.g., used to determine the probe timeout, etc.) that is different from a second algorithm (e.g., used to negotiate an idle timeout period between nodes, etc.) on which a determination of the second duration (e.g., duration associated with idle timeout in connection with 1-RTT packets, etc.) is based.

Claim	Claim Elements	Applicability
	from a second algorithm on which a determination of the second duration is based.	See excerpt(s) above and below, for example (emphasis added, if any): "10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout (see Section 18.1) and three times the current Probe Timeout (PTO). Each endpoint advertises its own idle timeout to its peer. An endpoint restarts any timer it maintains when a packet from its peer is received and processed successfully. The timer is also restarted when sending a packet containing frames other than ACK or PADDING (an ACK-eliciting packet; see [QUIC-RECOVERY]), but only if no other ACK-eliciting packets have been sent since last receiving a packet. Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. The value for an idle timeout can be asymmetric. The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint. An endpoint that sends packets near the end of the idle timeout period of a peer risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could timeout within a Probe Timeout (PTO; see Section 6.3 of [QUIC-RECOVERY]), it is advisable to test for liveness before sending any data that cannot be retried safely. Note that it is likely that only applications or application protocols will know what information can be retried." *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "18.1 Transport Parameter Definitions … • max_ack_delay (0x000b): The maximum ACK delay is an integer value indicating the maximum amount of time in milliseconds by which the endpoint will delay sending acknowledgments. This value SHOULD include the receiver's expected delays in alarms firing. For example, if a receiver sets a timer for 5ms and alarms commonly fire up to

Claim	Claim Elements	Applicability
		1ms late, then it should send a max_ack_delay of 6ms. If this value is absent, a default of 25 milliseconds is assumed. Values of 2^14 or greater are invalid" *QUIC: A UDP-Based Multiplexed and Secure Transport* https://tools.ietf.org/html/draft-ietf-quic-transport-22 "6.3. Probe Timeout A Probe Timeout (PTO) triggers a probe packet when ack-eliciting data is in flight but an acknowledgement is not received within the expected period of time. A PTO enables a connection to recover from loss of tail packets or acks. The PTO algorithm used in QUIC implements the reliability functions of Tail Loss Probe [TLP] [RACK], RTO [RFC5681] and F-RTO algorithms for TCP [RFC5682], and the timeout computation is based on TCP's retransmission timeout period [RFC6298]. 6.3.1. Computing PTO When an ack-eliciting packet is transmitted, the sender schedules a timer for the PTO period as follows: PTO = smoothed_rtt + max(4*rttvar, kGranularity) + max_ack_delay kGranularity, smoothed_rtt, rttvar, and max_ack_delay are defined in Appendix A.2 and Appendix A.3. The PTO period is the amount of time that a sender ought to wait for an acknowledgement of a sent packet. This time period includes the estimated network roundtrip-time (smoothed_rtt), the variance in the estimate (4*rttvar), and max_ack_delay, to account for the maximum time by which a receiver might delay sending an acknowledgement.

Claim	Claim Elements	Applicability
		The PTO value MUST be set to at least kGranularity, to avoid the timer expiring immediately. When a PTO timer expires, the sender probes the network as described in the next section. The PTO period MUST be set to twice its current value. This exponential reduction in the sender's rate is important because the PTOs might be caused by loss of packets or acknowledgements due to severe congestion. A sender computes its PTO timer every time an ack-eliciting packet is sent. A sender might choose to optimize this by setting the timer fewer times if it knows that more ack-eliciting packets will be sent within a short period of time" *QUIC Loss Detection and Congestion Control* https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-22#section-6.3

Caveat: The notes and/or cited excerpts utilized herein are set forth for illustrative purposes only and are not meant to be limiting in any manner. For example, the notes and/or cited excerpts, may or may not be supplemented or substituted with different excerpt(s) of the relevant reference(s), as appropriate. Further, to the extent any error(s) and/or omission(s) exist herein, all rights are reserved to correct the same.