

JENAM TECH LLC'S FIRST AMENDED SET OF INFRINGEMENT CONTENTIONS

U.S. Patent No. 10,742,774 – Google LLC

Claims 3, 6, 7, 10, 11, 15, 16, 17, 19, 32, 36, 37, 38, 39, 44, 51, 59, 65, 67, and 72

Jenam Tech LLC (“Jenam”) provides evidence of infringement of claims 3, 6, 7, 10, 11, 15, 16, 17, 19, 32, 36, 37, 38, 39, 44, 51, 59, 65, 67, and 72 of U.S. Patent No. 10,742,774 (hereinafter “the ’774 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 ’774 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 ’774 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 3, 6, 7, 10, 11, 15, 16, 17, 19, 32, 36, 37, 38, 39, 44, 51, 59, 65, 67, and 72 of the ’774 patent, including without limitation, the Accused Instrumentalities.

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

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 '774 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 1 Elements	Applicability
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 for: receiving, by a second node from a first node, a transmission control protocol (TCP)-variant packet in advance of a TCP-variant connection being established;	Google owns, controls, and/or uses an apparatus (e.g., one or more servers and/or the Google Cloud Platform, or at least one component thereof, etc.) 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 for: receiving, by a second node (e.g., the other one of the QUIC-compliant server or client, etc.) from a first node (e.g., one of a QUIC-compliant server or client, etc.), a transmission control protocol (TCP)-variant packet (e.g., a QUIC negotiation packet, etc.) in advance of a TCP-variant connection (e.g., a QUIC connection, etc.) being established. See excerpt(s) below, for example (emphasis added, if any): Note: Google admits that their "Google Cloud Platform" uses QUIC: "Google Cloud Platform Blog...

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	[W]e'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: 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 CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	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 CHARTS

BASED ON INFRINGEMENT ANALYSIS OF GOOGLE

U.S. Patent No. 10,742,774

Account

Search

Maps

YouTube

Play

News

Calendar

Gmail

Contacts

Drive

Translate

Photos

Cases overview

Alameda County

Confirmed1,191RecoveredDeaths42

California

google.com/search?safe=strict&xsrf=ALeKk00aNIUIhuNrpELxoFKiKChmEHwHqW:1587486895209&q=coronavirus+tips&oi=ddle&ct=153607016&hl=en&sa=X&ve...

Elements

Console

Sources

Network

Performance

Memory

Application

Security

Audits

Filter

Hide data URLs

All

XHR

JS

CSS

Img

Media

Font

Doc

WS

Manifest

Other

Only show requests with SameSite issues

1000 ms

2000 ms

3000 ms

4000 ms

5000 ms

6000 ms

7000 ms

8000 ms

9000 ms

Name	St...	Protocol	Domain	Type	Initiator	Size	Ti...	Waterfall
KfOmCnqku92Fr1...	200	http/2+quic/46	fonts.gstatic.com	font	app/origin=http...	(disk cac...	6 ...	
ListAccounts?listPa...	200	http/2+quic/46	accounts.google.com	document	rs=AA2YrTuismE...	1.4 KB	1...	
photo.jpg	200	http/2+quic/46	lh3.googleusercontent.c...	jpeg	rs=AA2YrTuismE...	(disk cac...	1...	
photo.jpg	200	http/2+quic/46	lh3.googleusercontent.c...	jpeg	rs=AA2YrTuismE...	(disk cac...	1...	
photo.jpg	200	http/2+quic/46	lh3.googleusercontent.c...	jpeg	rs=AA2YrTuismE...	(disk cac...	1...	
photo.jpg	200	http/2+quic/46	lh3.googleusercontent.c...	png	rs=AA2YrTuismE...	(disk cac...	8 ...	
m=wmwg8b	200	http/2+quic/46	www.gstatic.com	script	m= b_tp:434	(disk cac...	3 ...	
m=A7fCUjBVgquf,...	200	http/2+quic/46	www.gstatic.com	script	m= b_tp:434	(disk cac...	1...	
log?format=json&...	200	http/2+quic/46	play.google.com	xhr	m= b_tp:301	644 B	1...	
m=FCpbqb,OJUrv...	200	http/2+quic/46	www.gstatic.com	script	m= b_tp:434	(disk cac...	3 ...	
log?format=json&...	200	http/2+quic/46	play.google.com	xhr	m= b_tp:301	788 B	1...	
photo.jpg	200	http/2+quic/46	lh3.googleusercontent.c...	jpeg	search?safe=stria...	2.4 KB	1...	
photo.jpg	200	http/2+quic/46	lh3.googleusercontent.c...	jpeg	search?safe=stria...	2.2 KB	2...	
photo.jpg	200	http/2+quic/46	lh3.googleusercontent.c...	jpeg	search?safe=stria...	2.7 KB	1...	
photo.jpg	200	http/2+quic/46	lh3.googleusercontent.c...	png	search?safe=stria...	662 B	1...	
gen_204?atyp=i&e...	204	http/2+quic/46	www.google.com	text/html	rs=ACT90oFF1ed...	40 B	5...	
gen_204?atyp=csi...	204	http/2+quic/46	www.google.com	text/html	rs=ACT90oFF1ed...	40 B	5...	
log?format=json&...	200	http/2+quic/46	play.google.com	xhr	rs=AA2YrTuismE...	345 B	5...	

Note: As set forth below, QUIC is a “variant” of TCP.

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>

Note: The following evidence is related to and/or describes the QUIC standard contained herein.

5 of 81

March 17, 2021

Page 5 of 81

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	“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 Note: As set forth below, a QUIC connection establishment begins by sending a handshake packet. 7.2. Version Negotiation QUIC’s connection establishment begins with version negotiation, since all communication between the endpoints, including packet and frame formats, relies on the two endpoints agreeing on a version. A QUIC connection begins with a client sending a Client Initial packet (Section 5.4.1). The details of the handshake mechanisms are described in Section 7.3, but all of the initial packets sent from the client to the server MUST use the long header format - which includes the version of the protocol being used - and they MUST be padded to at least 1200 octets. The design of version negotiation permits a server to avoid maintaining state for packets that it rejects in this fashion. However, when the server generates a Version Negotiation packet, it cannot randomly generate a reserved version number. This is because the server is required to include the same value in its transport parameters (see Section 7.4.4). To avoid the selected version number o The initial handshake packet from the client needs to fit in a single packet (#338) QUIC packet: A well-formed UDP payload that can be parsed by a QUIC receiver. QUIC packet size in this document refers to the UDP <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/html/draft-ietf-quic-transport-07 Note: As set forth below, prior to a QUIC connection being established, the QUIC connection is “set up” using the aforementioned handshake.
--	---

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	3.1. Low-Latency Connection Establishment QUIC relies on a combined cryptographic and transport handshake for setting up a secure transport connection. QUIC connections are expected to commonly use 0-RTT handshakes, meaning that for most QUIC connections, data can be sent immediately following the client handshake packet, without waiting for a reply from the server. QUIC provides a dedicated stream (Stream ID 0) to be used for performing the cryptographic handshake and QUIC options negotiation. The format of the QUIC options and parameters used during negotiation are described in this document, but the handshake protocol that runs on Stream ID 0 is described in the accompanying cryptographic handshake draft [QUIC-TLS]. <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/html/draft-ietf-quic-transport-07
detecting a time period parameter field in the TCP-variant packet;	Google owns, controls, and/or uses the apparatus (e.g., the one or more servers and/or the Google Cloud Platform, or at least one component thereof, etc.) with the one or more processors that execute the instructions for: detecting a time period parameter field (e.g., idle timeout parameter field, etc.) in the TCP-variant packet (e.g., QUIC negotiation packet, etc.). See excerpt(s) 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. 7.4. 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 CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

Value	Parameter Name	Specification
0x0000	initial_max_stream_data	Section 7.4.1
0x0001	initial_max_data	Section 7.4.1
0x0002	initial_max_stream_id	Section 7.4.1
0x0003	idle_timeout	Section 7.4.1
0x0004	omit_connection_id	Section 7.4.1
0x0005	max_packet_size	Section 7.4.1
0x0006	stateless_reset_token	Section 7.4.1

Table 4: Initial QUIC Transport Parameters Entries

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	The format of the transport parameters is the TransportParameters struct from Figure 6. This is described using the presentation language from Section 3 of [I-D.ietf-tls-tls13]. <pre>uint32 QuicVersion; enum { initial_max_stream_data(0), initial_max_data(1), initial_max_stream_id(2), idle_timeout(3), omit_connection_id(4), max_packet_size(5), stateless_reset_token(6), (65535) } TransportParameterId;</pre> <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/html/draft-ietf-quic-transport-07
identifying metadata in the time period parameter field for a time period, during which no packet is received from the first node in a data stream of the TCP-variant connection and processed by the second node to keep the TCP-variant connection active; and	Google owns, controls, and/or uses the apparatus (e.g., the one or more servers and/or the Google Cloud Platform, or at least one component thereof, etc.) with the one or more processors that execute the instructions for: identifying metadata (e.g., a value of the idle timeout parameter field, etc.) in the time period parameter field (e.g., idle timeout parameter field, etc.) for a time period, during which no packet is received from the first node (e.g., one of the QUIC-compliant server or client, etc.) in a data stream of the TCP-variant connection (e.g., the QUIC connection, etc.) and processed by the second node (e.g., the other one of the QUIC-compliant server or client, etc.) to keep the TCP-variant connection active. See excerpt(s) below, for example (emphasis added, if any): Note: As set forth below, the metadata includes a value in seconds that is encoded as an unsigned 16-bit integer.

March 17, 2021

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	idle_timeout (0x0003): The idle timeout is a value in seconds that is encoded as an unsigned 16-bit integer. The maximum value is 600 seconds (10 minutes). 7.8.2. Idle Timeout A connection that remains idle for longer than the idle timeout (see Section 7.4.1) becomes closed. Either peer removes connection state if they have neither sent nor received a packet for this time. The time at which an idle timeout takes effect won't be perfectly synchronized on peers. A connection enters the draining period when the idle timeout expires. During this time, an endpoint that receives new packets MAY choose to restore the connection. Alternatively, an endpoint that receives packets MAY signal the timeout using an immediate close. <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/html/draft-ietf-quic-transport-07
calculating, by the second node and based on the metadata, a timeout attribute associated with the TCP-variant connection for being used to at least partially close the TCP-variant connection.	Google owns, controls, and/or uses the apparatus (e.g., the one or more servers and/or the Google Cloud Platform, or at least one component thereof, etc.) with the one or more processors that execute the instructions for: calculating, by the second node (e.g., the other one of the QUIC-compliant server or client, etc.) and based on the metadata (e.g., the value of the idle timeout parameter field, etc.), a timeout attribute associated with the TCP-variant connection (e.g., the QUIC connection, etc.) for being used to at least partially close the TCP-variant connection. See excerpt(s) 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 modified based on the value received in the idle_timeout field of the connection negotiation packet.

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	idle_timeout (0x0003): The idle timeout is a value in seconds that is encoded as an unsigned 16-bit integer. The maximum value is 600 seconds (10 minutes). 7.8. Connection Termination Connections should remain open until they become idle for a pre-negotiated period of time. A QUIC connection, once established, can be terminated in one of three ways: 7.8.2. Idle Timeout A connection that remains idle for longer than the idle timeout (see Section 7.4.1) becomes closed. Either peer removes connection state if they have neither sent nor received a packet for this time. The time at which an idle timeout takes effect won't be perfectly synchronized on peers. A connection enters the draining period when the idle timeout expires. During this time, an endpoint that receives new packets MAY choose to restore the connection. Alternatively, an endpoint that receives packets MAY signal the timeout using an immediate close. <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/html/draft-ietf-quic-transport-07 Note: The following evidence is related to and/or describes the QUIC standard contained herein. "5. Connection Closure Once established, an HTTP/3 connection can be used for many requests and responses over time until the connection is closed. <u>Connection closure can happen in any of several different ways.</u> 5.1. Idle Connections
--	--

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	<u>Each QUIC endpoint declares an idle timeout during the handshake. If the connection remains idle (no packets received) for longer than this duration, the peer will assume that the connection has been closed.</u> HTTP/3 implementations will need to open a new connection for new requests if the existing connection has been idle for longer than the server's advertised idle timeout, and SHOULD do so if approaching the idle timeout. HTTP clients are expected to request that the transport keep connections open while there are responses outstanding for requests or server pushes, as described in Section 10.2.2 of [QUIC-TRANSPORT]. <u>If the client is not expecting a response from the server, allowing an idle connection to time out is preferred over expending effort maintaining a connection that might not be needed.</u> A gateway MAY maintain connections in anticipation of need rather than incur the latency cost of connection establishment to servers. <u>Servers SHOULD NOT actively keep connections open.</u>" https://www.ietf.org/id/draft-ietf-quic-applicability-07.txt
--	--

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
 U.S. Patent No. 10,742,774

Claim 3 Elements	Applicability
The apparatus of claim 1 wherein the apparatus is configured such that, during the time period, no packet is received from the first node in the data stream of the TCP-variant connection regardless whether being processed by the second node.	Google infringes claim 1 and uses the apparatus (e.g., the one or more servers and/or the Google Cloud Platform, or at least one component thereof, etc.) wherein the apparatus is configured such that, during the time period, no packet is received from the first node (e.g., one of the QUIC-compliant server or client, etc.) in the data stream of the TCP-variant connection (e.g., the QUIC connection, etc.) regardless whether being processed by the second node (e.g., the other one of the QUIC-compliant server or client, etc.). See excerpt(s) below, for example (emphasis added, if any): “19.2. PING Frame <u>Endpoints can use PING frames (type=0x01) to verify that their peers are still alive or to check reachability to the peer.</u> 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. <u>A connection will time out if no packets are sent or received for a period longer than the time specified in the idle timeout transport parameter</u> (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.” <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/html/draft-ietf-quic-transport-22

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
 U.S. Patent No. 10,742,774

Claim 6 Elements	Applicability
The apparatus of claim 1 wherein the apparatus is configured such that the timeout attribute associated with the TCP-variant connection is capable of being used by the second node to unilaterally cause the TCP-variant connection to at least partially close.	Google infringes claim 1 and uses the apparatus (e.g., the one or more servers and/or the Google Cloud Platform, or at least one component thereof, etc.) wherein the apparatus is configured such that the timeout attribute associated with the TCP-variant connection (e.g., the QUIC connection, etc.) is capable of being used by the second node (e.g., the other one of the QUIC-compliant server or client, etc.) to unilaterally cause the TCP-variant connection to at least partially close. See excerpt(s) below, for example (emphasis added, if any): Note: As set forth below, after the idle time period is detected, the QUIC connection is closed due to inactivity. “10.2. Idle Timeout <u>If the idle timeout is enabled by either peer, a connection is silently closed and its state is discarded when it remains idle for longer than the minimum of the max_idle_timeouts (see Section 18.2) and three times the current Probe Timeout (PTO).</u> Each endpoint advertises a max_idle_timeout, but the effective value at an endpoint is computed as the minimum of the two advertised values. By announcing a max_idle_timeout, <u>an endpoint commits to initiating an immediate close (Section 10.3) if it abandons the connection prior to the effective value.</u>” <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/id/draft-ietf-quic-transport-27.txt

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

Claim 7 Elements	Applicability
The apparatus of claim 1 wherein the apparatus is configured such that, during the time period, no packet is received, nor expected to be received, from the first node in the data stream of the TCP-variant connection.	Google infringes claim 1 and uses the apparatus (e.g., the one or more servers and/or the Google Cloud Platform, or at least one component thereof, etc.) wherein the apparatus is configured such that, during the time period, no packet is received, nor expected to be received, from the first node (e.g., one of the QUIC-compliant server or client, etc.) in the data stream of the TCP-variant connection (e.g., the QUIC connection, etc.). See excerpt(s) below, for example (emphasis added, if any): “19.2. PING Frame <u>Endpoints can use PING frames (type=0x01) to verify that their peers are still alive or to check reachability to the peer.</u> 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. <u>A connection will time out if no packets are sent or received for a period longer than the time specified in the idle timeout transport parameter</u> (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.” <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/html/draft-ietf-quic-transport-22

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
 U.S. Patent No. 10,742,774

Claim 10 Elements	Applicability
The apparatus of claim 1 wherein the apparatus is configured such that a user timeout attribute, that is separate from the timeout attribute, is utilized for controlling communication of one or more acknowledgment packets.	Google infringes claim 1 and uses the apparatus (e.g., the one or more servers and/or the Google Cloud Platform, or at least one component thereof, etc.) wherein the apparatus is configured such that a user timeout attribute, that is separate from the timeout attribute, is utilized for controlling communication of one or more acknowledgment packets (e.g., responding to user timeout event within a transmission control block (TCB) designed to hold connection state information, etc.). See excerpt(s) below, for example (emphasis added, if any): Note: Based upon information and belief, the user timeout attribute in QUIC is based on the max_ack_delay transport parameter. See section 4.7 in the IETF QUIC recovery draft at https://tools.ietf.org/html/draft-ietf-quic-recovery-34#section-4.7. The definition of max_ack_delay in the QUIC transport establishes that the user timeout is used to control ACK packets (copied below). See also section 13.2.1 "Sending ACK Frames" in the IETF QUIC transport draft at https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-34#section-13.2.1. "max_ack_delay (0x0b): 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." https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-27#section-18.2 "13.2.1. Sending ACK Frames Every packet SHOULD be acknowledged at least once, and ack-eliciting packets MUST be acknowledged at least once within the maximum ack delay. An endpoint communicates its maximum delay using the max_ack_delay transport parameter; see Section 18.2. max_ack_delay declares an explicit contract: an endpoint promises to never intentionally delay acknowledgments of an ack-eliciting packet by more than the indicated value. If it does, any excess accrues to the RTT estimate and could result in spurious or delayed retransmissions from the peer. For Initial and Handshake packets, a max_ack_delay of 0 is used. The sender uses the receiver's "max_ack_delay"

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	value in determining timeouts for timer-based retransmission, as detailed in Section 5.2.1 of [QUIC-RECOVERY]." https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-27 Note: The following evidence is related to and/or describes the QUIC standard contained herein. “3.9. <u>Event Processing</u> The processing depicted in this section is an example of one possible implementation. Other implementations may have slightly different processing sequences, but they should differ from those in this section only in detail, not in substance. <u>The activity of the TCP can be characterized as responding to events.</u> The events that occur can be cast into three categories: user calls, arriving segments, and <u>timeouts</u>. This section describes the processing the TCP does in response to each of the events. In many cases the processing required depends on the state of the connection. ... Timeouts <u>USER TIMEOUT</u> RETRANSMISSION TIMEOUT TIME-WAIT TIMEOUT ... OPEN Call CLOSED STATE (i.e., TCB does not exist) <u>Create a new transmission control block (TCB) to hold connection state information. Fill in local socket identifier, foreign socket, precedence, security/compartments, and user timeout information.</u> Note that some parts of the foreign socket may be unspecified in a passive OPEN and are to be filled in by the parameters of the incoming SYN segment. Verify the security and precedence requested are allowed for this user, if not return "error: precedence not allowed" or "error: security/compartments not allowed." If passive enter the LISTEN state and return. If active and the
--	---

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	foreign socket is unspecified, return "error: foreign socket unspecified"; if active and the foreign socket is specified, issue a SYN segment. An initial send sequence number (ISS) is selected. A SYN segment of the form <SEQ=ISS><CTL=SYN> is sent. Set SND.UNA to ISS, SND.NXT to ISS+1, enter SYN-SENT state, and return." https://tools.ietf.org/html/rfc793 (emphasis added)
--	---

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
 U.S. Patent No. 10,742,774

Claim 11 Elements	Applicability
The apparatus of claim 10 wherein the apparatus is configured such that the user timeout attribute is determined based on the time period.	Google infringes claim 10 and uses the apparatus (e.g., the one or more servers and/or the Google Cloud Platform, or at least one component thereof, etc.) wherein the apparatus is configured such that the user timeout attribute is determined based on the time period. Note: Based upon information and belief, the user timeout in QUIC is based on the max_ack_delay transport parameter negotiated during setup. It is utilized to determine a probe timeout (PTO). Note that the max_ack_delay parameter is negotiated during setup. See QUIC transport draft section 13.2.1 at Hypertext Transfer Protocol Version 3 (HTTP/3) (ietf.org), section 4.7 in the IETF QUIC recovery draft at https://tools.ietf.org/html/draft-ietf-quic-recovery-34#section-4.7, and section 10.1 in the IETF QUIC transport draft at https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-34#section-10.1. Note: The following evidence is related to and/or describes the QUIC standard contained herein. “Case 2: TCP receives a FIN from the network If an unsolicited FIN arrives from the network, the receiving TCP can ACK it and tell the user that the connection is closing. The user will respond with a CLOSE, upon which the TCP can send a FIN to the other TCP after sending any remaining data. <u>The TCP then waits until its own FIN is acknowledged whereupon it deletes the connection. If an ACK is not forthcoming, after the user timeout the connection is aborted and the user is told.</u> ... 3.9. <u>Event Processing</u> The processing depicted in this section is an example of one possible implementation. Other implementations may have slightly different processing sequences, but they should differ from those in this section only in detail, not in substance. <u>The activity of the TCP can be characterized as responding to events.</u> The events that occur can be cast into three categories: user calls, arriving segments, and <u>timeouts</u>. This section describes the processing the TCP does in response to each of the events. In many cases the processing required depends on the state of the connection.

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	... Timeouts <u>USER TIMEOUT</u> RETRANSMISSION TIMEOUT TIME-WAIT TIMEOUT ... OPEN Call CLOSED STATE (i.e., TCB does not exist) <u>Create a new transmission control block (TCB) to hold connection state information. Fill in local socket identifier, foreign socket, precedence, security/compartments, and user timeout information.</u> Note that some parts of the foreign socket may be unspecified in a passive OPEN and are to be filled in by the parameters of the incoming SYN segment. Verify the security and precedence requested are allowed for this user, if not return "error: precedence not allowed" or "error: security/compartments not allowed." If passive enter the LISTEN state and return. If active and the foreign socket is unspecified, return "error: foreign socket unspecified"; if active and the foreign socket is specified, issue a SYN segment. An initial send sequence number (ISS) is selected. A SYN segment of the form <SEQ=ISS><CTL=SYN> is sent. Set SND.UNA to ISS, SND.NXT to ISS+1, enter SYN-SENT state, and return." https://tools.ietf.org/html/rfc793 (emphasis added)
--	--

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
 U.S. Patent No. 10,742,774

Claim 15 Elements	Applicability
The apparatus of claim 1 wherein the apparatus is configured such that the TCP-variant connection is at least partially closed without any keep-alive packet caused to be communicated before the TCP-variant connection is at least partially closed.	Google infringes claim 1 and uses the apparatus (e.g., the one or more servers and/or the Google Cloud Platform, or at least one component thereof, etc.) wherein the apparatus is configured such that the TCP-variant connection (e.g., the QUIC connection, etc.) is at least partially closed without any keep-alive packet caused to be communicated before the TCP-variant connection is at least partially closed. See excerpt(s) below, for example (emphasis added, if any): Note: As set forth below, after the idle time period is detected, the QUIC connection is closed due to inactivity. “10.2. Idle Timeout If the idle timeout is enabled by either peer, a connection is silently closed and its state is discarded when it remains idle for longer than the minimum of the max_idle_timeouts (see Section 18.2) and three times the current Probe Timeout (PTO).” <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/id/draft-ietf-quic-transport-27.txt “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.

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	<u>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). . .</u> <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/id/draft-ietf-quic-transport-27.txt Note: The following evidence is related to and/or describes the QUIC standard contained herein. "5. Connection Closure Once established, an HTTP/3 connection can be used for many requests and responses over time until the connection is closed. <u>Connection closure can happen in any of several different ways.</u> 5.1. Idle Connections <u>Each QUIC endpoint declares an idle timeout during the handshake. If the connection remains idle (no packets received) for longer than this duration, the peer will assume that the connection has been closed.</u> HTTP/3 implementations will need to open a new connection for new requests if the existing connection has been idle for longer than the server's advertised idle timeout, and SHOULD do so if approaching the idle timeout. HTTP clients are expected to request that the transport keep connections open while there are responses outstanding for requests or server pushes, as described in Section 10.2.2 of [QUIC-TRANSPORT]. <u>If the client is not expecting a response from the server, allowing an idle connection to time out is preferred over expending effort maintaining a connection that might not be needed.</u> A gateway MAY maintain connections in anticipation of need rather than incur the latency cost of connection establishment to servers. <u>Servers SHOULD NOT actively keep connections open.</u>" https://www.ietf.org/id/draft-ietf-quic-applicability-07.txt Note: The following evidence is related to and/or describes the QUIC standard contained herein. "3.3. Session resumption versus Keep-alive
--	---

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	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. The first strategy is the easiest, but it only applies to certain applications. Either the server or the client in a QUIC application can send PING frames as keep-alives, to prevent the connection and any on-path state from timing out. <u>Recommendations for the use of keep-alives are application specific, mainly depending on the latency requirements and message frequency of the application. In this case, the application mapping must specify whether the client or server is responsible for keeping the application alive.</u> Note that sending PING frames more frequently than every 30 seconds over long idle periods may result in a too much unproductive traffic and power usage for some situations. <u>Alternatively, the client (but not the server) can use session resumption instead of sending keepalive traffic.</u> 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. The tradeoffs between resumption and keepalive need to be evaluated on a per-application basis. However, in general applications should use keepalives only in circumstances where continued
--	--

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	communication is highly likely; [QUIC-HTTP], for instance, recommends using PING frames for keepalive only when a request is outstanding.” https://www.ietf.org/id/draft-ietf-quic-applicability-07.txt
--	---

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
 U.S. Patent No. 10,742,774

Claim 16 Elements	Applicability
The apparatus of claim 1 wherein the apparatus is configured such that a keep-alive attribute, that is separate from the timeout attribute, is utilized for controlling communication of one or more keep-alive packets before an expiration of the time period.	Google infringes claim 1 and uses the apparatus (e.g., the one or more servers and/or the Google Cloud Platform, or at least one component thereof, etc.) wherein the apparatus is configured such that a keep-alive attribute, that is separate from the timeout attribute, is utilized for controlling communication of one or more keep-alive packets before an expiration of the time period. See excerpt(s) below, for example (emphasis added, if any): Note: Based upon information and belief, infringement occurs, for example, at least during a packet communication scenario when a keep-alive packet starts a detected idle time period and no other keep-alives are communicated during the idle period. 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 created or 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.” <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> 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).” <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/html/draft-ietf-quic-transport-22

March 17, 2021

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	“10.2. Idle Timeout If the idle timeout is enabled, a connection is silently closed and the state is discarded when it <u>remains idle for longer than both the advertised idle timeout</u> (see Section 18.1) and three times the current Probe Timeout (PTO).” “The value for an idle timeout can be asymmetric. <u>The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint.</u> 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.” <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> 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 <u>peers are still alive or to check reachability to the peer.</u> <u>The PING frame contains no additional fields.</u> The <u>receiver of a PING frame simply needs to acknowledge the packet</u> 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 specified in the 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
--	--

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

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

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

Claim 17 Elements	Applicability
The apparatus of claim 16 wherein the apparatus is configured such that the keep-alive attribute is calculated based on the time period.	Google infringes claim 16 and uses the apparatus (e.g., the one or more servers and/or the Google Cloud Platform, or at least one component thereof, etc.) wherein the apparatus is configured such that the keep-alive attribute is calculated based on the time period. See excerpt(s) below, for example (emphasis added, if any): “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.” <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/html/draft-ietf-quic-transport-22 “10.2. Idle Timeout If the idle timeout is enabled by either peer, a connection is silently closed and its state is discarded when it remains idle for longer than the minimum of the max_idle_timeouts (see Section 18.2) and three times the current Probe Timeout (PTO).” <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/html/draft-ietf-quic-transport-27

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
 U.S. Patent No. 10,742,774

Claim 18 Elements	Applicability
The apparatus of claim 1 wherein the apparatus is configured such that the timeout attribute is used to at least partially close the TCP-variant connection, by the TCP-variant connection being at least partially closed based on the timeout attribute.	Google infringes claim 1 and uses the apparatus (e.g., the one or more servers and/or the Google Cloud Platform, or at least one component thereof, etc.) wherein the apparatus is configured such that the timeout attribute is used to at least partially close the TCP-variant connection (e.g., the QUIC connection, etc.), by the TCP-variant connection being at least partially closed based on the timeout attribute. See excerpt(s) below, for example (emphasis added, if any): Note: As set forth below, after the idle time period is detected, the QUIC connection is closed due to inactivity. “10.2. Idle Timeout If the idle timeout is enabled by either peer, a connection is silently closed and its state is discarded when it remains idle for longer than the minimum of the max_idle_timeouts (see Section 18.2) and three times the current Probe Timeout (PTO). <u>Each endpoint advertises a max_idle_timeout, but the effective value at an endpoint is computed as the minimum of the two advertised values.</u> By announcing a max_idle_timeout, an endpoint commits to initiating an immediate close (Section 10.3) if it abandons the connection prior to the effective value. An endpoint restarts its idle timer when a packet from its peer is received and processed successfully. The idle timer is also restarted when sending 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. An endpoint might need to send packets to avoid an idle timeout if it is unable to send application data due to being blocked on flow control limits; see Section 4. <u>An endpoint that sends packets near the end of the idle timeout period risks having those packets discarded if its peer enters the draining state before the packets arrive.</u> If a peer could time out within a Probe Timeout (PTO; see Section 6.6 of [QUIC-RECOVERY]), it is advisable to test for liveness

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	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.” <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/id/draft-ietf-quic-transport-27.txt Note: The following evidence is related to and/or describes the QUIC standard contained herein. "5. Connection Closure Once established, an HTTP/3 connection can be used for many requests and responses over time until the connection is closed. <u>Connection closure can happen in any of several different ways.</u> 5.1. Idle Connections <u>Each QUIC endpoint declares an idle timeout during the handshake. If the connection remains idle (no packets received) for longer than this duration, the peer will assume that the connection has been closed.</u> HTTP/3 implementations will need to open a new connection for new requests if the existing connection has been idle for longer than the server's advertised idle timeout, and SHOULD do so if approaching the idle timeout. HTTP clients are expected to request that the transport keep connections open while there are responses outstanding for requests or server pushes, as described in Section 10.2.2 of [QUIC-TRANSPORT]. <u>If the client is not expecting a response from the server, allowing an idle connection to time out is preferred over expending effort maintaining a connection that might not be needed.</u> A gateway MAY maintain connections in anticipation of need rather than incur the latency cost of connection establishment to servers. <u>Servers SHOULD NOT actively keep connections open.</u>" https://www.ietf.org/id/draft-ietf-quic-applicability-07.txt
--	--

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
 U.S. Patent No. 10,742,774

Claim 19 Elements	Applicability
The apparatus of claim 18 wherein the apparatus is configured such that the TCP-variant connection is at least partially closed, by the second node, without sending any packet for delivery to the first node during the time period.	Google infringes claim 18 and uses the apparatus (e.g., the one or more servers and/or the Google Cloud Platform, or at least one component thereof, etc.) wherein the apparatus is configured such that the TCP-variant connection (e.g., the QUIC connection, etc.) is at least partially closed, by the second node (e.g., the other one of the QUIC-compliant server or client, etc.), without sending any packet for delivery to the first node (e.g., one of the QUIC-compliant server or client, etc.) during the time period. See excerpt(s) below, for example (emphasis added, if any): Note: As set forth below, after the idle time period is detected, the QUIC connection is closed due to inactivity. “10.2. Idle Timeout If the idle timeout is enabled by either peer, a connection is silently closed and its state is discarded when it remains idle for longer than the minimum of the max_idle_timeouts (see Section 18.2) and three times the current Probe Timeout (PTO). <u>Each endpoint advertises a max_idle_timeout, but the effective value at an endpoint is computed as the minimum of the two advertised values.</u> By announcing a max_idle_timeout, an endpoint commits to initiating an immediate close (Section 10.3) if it abandons the connection prior to the effective value. An endpoint restarts its idle timer when a packet from its peer is received and processed successfully. The idle timer is also restarted when sending 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. An endpoint might need to send packets to avoid an idle timeout if it is unable to send application data due to being blocked on flow control limits; see Section 4. <u>An endpoint that sends packets near the end of the idle timeout period risks having those packets discarded if its peer enters the draining state before the packets arrive.</u> If a peer could time out

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	within a Probe Timeout (PTO; see Section 6.6 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.” <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/id/draft-ietf-quic-transport-27.txt “19.2. PING Frame Endpoints <u>can use</u> 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. <u>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).</u> 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. <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/id/draft-ietf-quic-transport-27.txt
--	--

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
 U.S. Patent No. 10,742,774

Claim 32 Elements	Applicability
The apparatus of claim 1 wherein the apparatus is configured such that at least an acknowledgment timeout is based on the metadata.	Google infringes claim 1 and uses the apparatus (e.g., the one or more servers and/or the Google Cloud Platform, or at least one component thereof, etc.) wherein the apparatus is configured such that at least an acknowledgment timeout (e.g., idle timeout, etc.) is based on the metadata. See excerpt(s) 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 <u>receiver of a PING frame simply needs to acknowledge the packet containing this frame.</u> The <u>PING frame can be used to keep a connection alive when an application or application protocol wishes to prevent the connection from timing out.</u> 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 specified in the idle_timeout transport parameter (see Section 10). However, state in middleboxes might time out earlier than that. Though REQ-5 in [RFC4787] <u>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.</u>” <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/html/draft-ietf-quic-transport-22

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
 U.S. Patent No. 10,742,774

Claim 36 Elements	Applicability
The apparatus of claim 1 wherein the apparatus is configured such that: the timeout attribute associated with the TCP-variant connection is capable of being used by the second node to unilaterally at least partially close the TCP-variant connection, without sending any packet for delivery to the first node during nor after the time period, by the TCP-variant connection being at least partially closed based on the timeout attribute and at least one of a formula, an expression, a function, or a policy;	Google infringes claim 1 and uses the apparatus (e.g., the one or more servers and/or the Google Cloud Platform, or at least one component thereof, etc.) wherein the apparatus is configured such that: the timeout attribute associated with the TCP-variant connection is capable of being used by the second node to unilaterally at least partially close the TCP-variant connection (e.g., the QUIC connection, etc.), without sending any packet for delivery to the first node during nor after the time period, by the TCP-variant connection being at least partially closed based on the timeout attribute and at least one of a formula, an expression, a function, or a policy (e.g., based on the idle_timeout duration, etc.). See excerpt(s) 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 determined based on the value received in the idle_timeout field of the connection negotiation packet. “10.2. Idle Timeout If the idle timeout is enabled by either peer, a connection is silently closed and its state is discarded when it remains idle for longer than the minimum of the max_idle_timeouts (see Section 18.2) and three times the current Probe Timeout (PTO). <u>Each endpoint advertises a max_idle_timeout, but the effective value at an endpoint is computed as the minimum of the two advertised values.</u> By announcing a max_idle_timeout, an endpoint commits to initiating an immediate close (Section 10.3) if it abandons the connection prior to the effective value. An endpoint restarts its idle timer when a packet from its peer is received and processed successfully. The idle timer is also restarted when sending 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. An endpoint might need to send packets to avoid an idle timeout if it is unable to send application data due to being blocked on flow control limits; see Section 4. <u>An endpoint that sends packets near the end of the idle timeout period risks having those packets discarded if its peer enters the draining state before the packets arrive.</u> If a peer could time out within a

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	Probe Timeout (PTO; see Section 6.6 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.” <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/id/draft-ietf-quic-transport-27.txt “19.2. PING Frame Endpoints <u>can use</u> 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. <u>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).</u> 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. <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/id/draft-ietf-quic-transport-27.txt Note: The following evidence is related to and/or describes the QUIC standard contained herein. "5. Connection Closure Once established, an HTTP/3 connection can be used for many requests and responses over time until the connection is closed. <u>Connection closure can happen in any of several different ways.</u>
--	--

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	5.1. Idle Connections <u>Each QUIC endpoint declares an idle timeout during the handshake. If the connection remains idle (no packets received) for longer than this duration, the peer will assume that the connection has been closed.</u> HTTP/3 implementations will need to open a new connection for new requests if the existing connection has been idle for longer than the server's advertised idle timeout, and SHOULD do so if approaching the idle timeout. HTTP clients are expected to request that the transport keep connections open while there are responses outstanding for requests or server pushes, as described in Section 10.2.2 of [QUIC-TRANSPORT]. <u>If the client is not expecting a response from the server, allowing an idle connection to time out is preferred over expending effort maintaining a connection that might not be needed.</u> A gateway MAY maintain connections in anticipation of need rather than incur the latency cost of connection establishment to servers. <u>Servers SHOULD NOT actively keep connections open.</u>" https://www.ietf.org/id/draft-ietf-quic-applicability-07.txt
the TCP-variant packet is not a synchronize (SYN) packet; and	Google infringes claim 1 and uses the apparatus (e.g., the one or more servers and/or the Google Cloud Platform, or at least one component thereof, etc.) wherein the TCP-variant packet (e.g., QUIC negotiation packet, etc.) is not a synchronize (SYN) packet. See excerpt(s) below, for example (emphasis added, if any): <u>Note:</u> As set forth below, synchronization is required between the two sides prior to establishing a TCP-variant connection (e.g., a QUIC connection, etc.).

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	7.2. Version Negotiation QUIC's connection establishment begins with version negotiation, since all communication between the endpoints, including packet and frame formats, relies on the two endpoints agreeing on a version. A QUIC connection begins with a client sending a Client Initial packet (Section 5.4.1). The details of the handshake mechanisms are described in Section 7.3, but all of the initial packets sent from the client to the server MUST use the long header format - which includes the version of the protocol being used - and they MUST be padded to at least 1200 octets. <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/html/draft-ietf-quic-transport-07 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." <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/html/draft-ietf-quic-transport-22
the TCP-variant connection is not a standard TCP connection.	Google infringes claim 1 and uses the apparatus (e.g., the one or more servers and/or the Google Cloud Platform, or at least one component thereof, etc.) wherein the TCP-variant connection (e.g., QUIC connection, etc.) is not a standard TCP connection. See excerpt(s) below, for example (emphasis added, if any): Note: Below is a web page of Google (https://www.google.com/).

U.S. Patent No. 10,742,774

The screenshot shows a Google search for a specific URL containing the parameter 'safe=strict'. The search results are not visible, but the network waterfall chart is open, showing various requests. The 'safe=strict' parameter is highlighted in the search bar. The waterfall chart shows requests to fonts.gstatic.com, accounts.google.com, and photos.gstatic.com. The 'safe=strict' parameter is highlighted in the search bar.

Note: As set forth below, QUIC is a “variant” of TCP.

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>

Note: The following evidence is related to and/or describes the QUIC standard contained herein.

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	“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
--	--

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
 U.S. Patent No. 10,742,774

Claim 37 Elements	Applicability
The apparatus of claim 36 wherein the apparatus is configured such that: the time period starts in response to the second node sending a keep-alive packet in another data stream of the TCP-variant connection for routing towards the first node; and	Google infringes claim 36 and uses the apparatus (e.g., the one or more servers and/or the Google Cloud Platform, or at least one component thereof, etc.) wherein the apparatus is configured such that: the time period starts in response to the second node (e.g., the other one of the QUIC-compliant server or client, etc.) sending a keep-alive packet in another data stream of the TCP-variant connection for routing towards the first node (e.g., one of the QUIC-compliant server or client, etc.). See excerpt(s) below, for example (emphasis added, if any): Note: Based upon information and belief, infringement occurs, for example, at least during a packet communication scenario when a keep-alive packet starts a detected idle time period and no other keep-alives are communicated during the idle period. Note: Based upon information and belief, the “second node” term is infringed by at least a server, and the “first node” term is infringed by at least a client. Note: As set forth below, a QUIC connection establishment begins by sending a handshake packet. “10.2. Idle Timeout . . . An endpoint restarts its idle timer when a packet from its peer is received and processed successfully. The idle timer is also restarted when sending 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. An endpoint might need to send packets to avoid an idle timeout if it is unable to send application data due to being blocked on flow control limits; see Section 4.” <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/id/draft-ietf-quic-transport-27.txt “19.2. PING Frame

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
 U.S. Patent No. 10,742,774

	Endpoints <u>can use</u> 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. <u>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.</u> 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. <u>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</u> (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. <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/id/draft-ietf-quic-transport-27.txt
during the time period, another keep-alive packet is sent from the second node in the another data stream of the TCP-variant connection for routing towards the first node, without restarting the time period.	Google infringes claim 36 and uses the apparatus (e.g., the one or more servers and/or the Google Cloud Platform, or at least one component thereof, etc.) wherein, during the time period, another keep-alive packet is sent from the second node (e.g., the other one of the QUIC-compliant server or client, etc.) in the another data stream of the TCP-variant connection (e.g., the QUIC connection, etc.) for routing towards the first node (e.g., one of the QUIC-compliant server or client, etc.), without restarting the time period. See excerpt(s) below, for example (emphasis added, if any): “10.2. Idle Timeout . . . <u>An endpoint restarts its idle timer when a packet from its peer is received and processed successfully. The idle timer is also restarted when sending an ack-eliciting packet (see [QUIC-RECOVERY]), but only if no other ack-eliciting packets have been sent since last receiving a packet.</u>

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. An endpoint might need to send packets to avoid an idle timeout if it is unable to send application data due to being blocked on flow control limits; see Section 4.” <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/id/draft-ietf-quic-transport-27.txt “19.2. PING Frame Endpoints <u>can use</u> 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. <u>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).</u> 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. <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/id/draft-ietf-quic-transport-27.txt
--	--

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
 U.S. Patent No. 10,742,774

Claim 38 Elements	Applicability
The apparatus of claim 37 wherein the apparatus is configured such that: a user timeout attribute, that is separate from the timeout attribute, is utilized for monitoring communication of one or more acknowledgment packets, where the user timeout attribute is determined based on the time period.	Google infringes claim 37 and uses the apparatus (e.g., the one or more servers and/or the Google Cloud Platform, or at least one component thereof, etc.) wherein the apparatus is configured such that: a user timeout attribute, that is separate from the timeout attribute, is utilized for monitoring communication of one or more acknowledgment packets, where the user timeout attribute is determined based on the time period. Note: Based upon information and belief, the user timeout in QUIC is based on the max_ack_delay transport parameter negotiated during setup. It is utilized to determine a probe timeout (PTO). Note that the max_ack_delay parameter is negotiated during setup. See QUIC transport draft section 13.2.1 at Hypertext Transfer Protocol Version 3 (HTTP/3) (ietf.org), section 4.7 in the IETF QUIC recovery draft at https://tools.ietf.org/html/draft-ietf-quic-recovery-34#section-4.7, and section 10.1 in the IETF QUIC transport draft at https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-34#section-10.1. Note: Based upon information and belief, the user timeout attribute in QUIC is based on the max_ack_delay transport parameter. See section 4.7 in the IETF QUIC recovery draft at https://tools.ietf.org/html/draft-ietf-quic-recovery-34#section-4.7. The definition of max_ack_delay in the QUIC transport establishes that the user timeout is used to control ACK packets (copied below). See also section 13.2.1 "Sending ACK Frames" in the IETF QUIC transport draft at https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-34#section-13.2.1. "max_ack_delay (0x0b): 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 2¹⁴ or greater are invalid." https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-27#section-18.2 "13.2.1. Sending ACK Frames

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	Every packet SHOULD be acknowledged at least once, and ack-eliciting packets MUST be acknowledged at least once within the maximum ack delay. An endpoint communicates its maximum delay using the max_ack_delay transport parameter; see Section 18.2. max_ack_delay declares an explicit contract: an endpoint promises to never intentionally delay acknowledgments of an ack-eliciting packet by more than the indicated value. If it does, any excess accrues to the RTT estimate and could result in spurious or delayed retransmissions from the peer. For Initial and Handshake packets, a max_ack_delay of 0 is used. The sender uses the receiver's "max_ack_delay" value in determining timeouts for timer-based retransmission, as detailed in Section 5.2.1 of [QUIC-RECOVERY]." https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-27 Note: The following evidence is related to and/or describes the QUIC standard contained herein. "Case 2: TCP receives a FIN from the network If an unsolicited FIN arrives from the network, the receiving TCP can ACK it and tell the user that the connection is closing. The user will respond with a CLOSE, upon which the TCP can send a FIN to the other TCP after sending any remaining data. <u>The TCP then waits until its own FIN is acknowledged whereupon it deletes the connection. If an ACK is not forthcoming, after the user timeout the connection is aborted and the user is told.</u> ... 3.9. <u>Event Processing</u> The processing depicted in this section is an example of one possible implementation. Other implementations may have slightly different processing sequences, but they should differ from those in this section only in detail, not in substance. <u>The activity of the TCP can be characterized as responding to events.</u> The events that occur can be cast into three categories: user calls, arriving segments, and <u>timeouts</u>. This section describes the processing the TCP does in response to each of the events. In many cases the processing required depends on the state of the connection. ... Timeouts
--	---

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	<u>USER TIMEOUT</u> RETRANSMISSION TIMEOUT TIME-WAIT TIMEOUT ... OPEN Call CLOSED STATE (i.e., TCB does not exist) <u>Create a new transmission control block (TCB) to hold connection state information. Fill in local socket identifier, foreign socket, precedence, security/compartment, and user timeout information.</u> Note that some parts of the foreign socket may be unspecified in a passive OPEN and are to be filled in by the parameters of the incoming SYN segment. Verify the security and precedence requested are allowed for this user, if not return "error: precedence not allowed" or "error: security/compartment not allowed." If passive enter the LISTEN state and return. If active and the foreign socket is unspecified, return "error: foreign socket unspecified"; if active and the foreign socket is specified, issue a SYN segment. An initial send sequence number (ISS) is selected. A SYN segment of the form <SEQ=ISS><CTL=SYN> is sent. Set SND.UNA to ISS, SND.NXT to ISS+1, enter SYN-SENT state, and return." https://tools.ietf.org/html/rfc793 (emphasis added)
--	---

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
 U.S. Patent No. 10,742,774

Claim 39 Elements	Applicability
The apparatus of claim 37 wherein the apparatus is configured such that: a keep-alive attribute, that is separate from the timeout attribute, is utilized for controlling communication of one or more keep-alive packets before an expiration of the time period, where the keep-alive attribute is determined based on the time period.	Google infringes claim 37 and uses the apparatus (e.g., the one or more servers and/or the Google Cloud Platform, or at least one component thereof, etc.) wherein the apparatus is configured such that: a keep-alive attribute, that is separate from the timeout attribute, is utilized for controlling communication of one or more keep-alive packets before an expiration of the time period, where the keep-alive attribute is determined based on the time period. See excerpt(s) 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 created or 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.” <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> 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).” <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/html/draft-ietf-quic-transport-22 “10.2. Idle Timeout

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	<u>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)."</u> "The value for an idle timeout can be asymmetric. <u>The value advertised by an endpoint is only used to determine whether the connection is live at that endpoint.</u> 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." <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> 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 <u>peers are still alive or to check reachability to the peer.</u> <u>The PING frame contains no additional fields.</u> The <u>receiver of a PING frame simply needs to acknowledge the packet</u> 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. <u>A connection will time out if no packets are sent or received for a period longer than the time specified in the idle_timeout transport parameter (see Section 10).</u> 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 CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

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

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
 U.S. Patent No. 10,742,774

Claim 42 Elements	Applicability
A method comprising: using at least a portion of a first node: receiving information for use in: detecting a time period that results in a non-TCP connection being subject to at least partial deactivation, and sending a non-TCP packet that is based on the information and that includes a time period parameter field identifying metadata; and	Google, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform that performs a method comprising: using at least a portion of a first node (e.g., one of a QUIC-compliant server or client, etc.): receiving information for use in: detecting a time period that results in a non-TCP connection (e.g., QUIC connection, etc.) being subject to at least partial deactivation (e.g., terminating the connection, etc.), and sending a non-TCP packet that is based on the information and that includes a time period parameter field identifying metadata (e.g., a value, etc.). See excerpt(s) below, for example (emphasis added, if any): Note: Google admits that their “Google Cloud Platform” uses QUIC: "Google Cloud Platform Blog... [W]e’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: https://tools.ietf.org/html/draft-ietf-quic-transport-00

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	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 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
--	--

March 17, 2021

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

(<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/>).

The screenshot shows a Google search page with the Network tab open in the developer tools. The Network tab displays a list of requests, with the 'Protocol' column highlighted in red. The highlighted protocols are 'http/2+quic/46'. The table below shows the details of the requests.

Name	Status	Protocol	Domain	Type	Initiator	Size	Time	Waterfall
KFUmCnqEu92Fr1...	200	http/2+quic/46	tonts.gstatic.com	font	app?origin=http...	(disk cac...	6 ...	
ListAccounts?listPa...	200	http/2+quic/46	accounts.google.com	document	rs=AA2YrTuismE...	1.4 KB	1...	
photo.jpg	200	http/2+quic/46	lh3.googleusercontent.c...	jpeg	rs=AA2YrTuismE...	(disk cac...	1...	
photo.jpg	200	http/2+quic/46	lh3.googleusercontent.c...	jpeg	rs=AA2YrTuismE...	(disk cac...	1...	
photo.jpg	200	http/2+quic/46	lh3.googleusercontent.c...	jpeg	rs=AA2YrTuismE...	(disk cac...	1...	
photo.jpg	200	http/2+quic/46	lh3.googleusercontent.c...	png	rs=AA2YrTuismE...	(disk cac...	8 ...	
m=wmwg8b	200	http/2+quic/46	www.gstatic.com	script	m= b_tp434	(disk cac...	3 ...	
m=A7fCU,BVgqf...	200	http/2+quic/46	www.gstatic.com	script	m= b_tp434	(disk cac...	1...	
log?format=json&...	200	http/2+quic/46	play.google.com	xhr	m= b_tp301	644 B	1...	
m=FCpbqb,OJUrv...	200	http/2+quic/46	www.gstatic.com	script	m= b_tp434	(disk cac...	3 ...	
log?format=json&...	200	http/2+quic/46	play.google.com	xhr	m= b_tp301	788 B	1...	
photo.jpg	200	http/2+quic/46	lh3.googleusercontent.c...	jpeg	search?safe=stria...	2.4 KB	1...	
photo.jpg	200	http/2+quic/46	lh3.googleusercontent.c...	jpeg	search?safe=stria...	2.2 KB	2...	
photo.jpg	200	http/2+quic/46	lh3.googleusercontent.c...	jpeg	search?safe=stria...	2.7 KB	1...	
photo.jpg	200	http/2+quic/46	lh3.googleusercontent.c...	png	search?safe=stria...	662 B	1...	
gen_204?atyp=i&e...	204	http/2+quic/46	www.google.com	text/html	rs=ACT90oFF1ed...	40 B	5...	
gen_204?atyp=csi...	204	http/2+quic/46	www.google.com	text/html	rs=ACT90oFF1ed...	40 B	5...	
log?format=json&...	200	http/2+quic/46	play.google.com	xhr	rs=AA2YrTuismE...	345 B	5...	

Note: As set forth below, QUIC is a “variant” of TCP.

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	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 Note: The following evidence is related to and/or describes the QUIC standard contained herein. “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 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.” <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> 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).” <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/html/draft-ietf-quic-transport-22 “10.2. Idle Timeout
--	--

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
 U.S. Patent No. 10,742,774

	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)." "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." <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/html/draft-ietf-quic-transport-22
sending, to a second node and for setting up the non-TCP connection, the non-TCP packet to provide the metadata to the second node, for use by the second node in determining a timeout attribute associated with the non-TCP connection.	Google, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform that performs a method including sending, to a second node (e.g., the other one of the QUIC-compliant server or client, etc.) and for setting up the non-TCP connection (e.g., QUIC connection, etc.), the non-TCP packet to provide the metadata (e.g., the value, etc.) to the second node, for use by the second node in determining a timeout attribute associated with the non-TCP connection. See excerpt(s) 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." <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/html/draft-ietf-quic-transport-22

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
 U.S. Patent No. 10,742,774

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 CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	“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]. <pre>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;</pre> <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/html/draft-ietf-quic-transport-22 Note: As set forth below, the metadata includes a value in seconds that is encoded as an unsigned 16-bit integer. “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.” <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/html/draft-ietf-quic-transport-22
--	--

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

Claim 43 Elements	Applicability
The method of claim 42 wherein: the determination of the timeout attribute results from a negotiation with the second node;	Google infringes claim 42 and, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform that performs a method wherein: the determination of the timeout attribute results from a negotiation with the second node (e.g., the other one of the QUIC-compliant server or client, etc.). See excerpt(s) 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.” <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/html/draft-ietf-quic-transport-22

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
 U.S. Patent No. 10,742,774

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 CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

“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 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 CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	“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).” <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/html/draft-ietf-quic-transport-22 “10.2. Idle Timeout If the idle timeout is enabled by either peer, a connection is silently closed and its state is discarded when it remains idle for longer than the minimum of the max_idle_timeouts (see Section 18.2) and three times the current Probe Timeout (PTO). <u>Each endpoint advertises a max_idle_timeout, but the effective value at an endpoint is computed as the minimum of the two advertised values.</u> By announcing a max_idle_timeout, an endpoint commits to initiating an immediate close (Section 10.3) if it abandons the connection prior to the effective value. An endpoint restarts its idle timer when a packet from its peer is received and processed successfully. The idle timer is also restarted when sending 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. An endpoint might need to send packets to avoid an idle timeout if it is unable to send application data due to being blocked on flow control limits; see Section 4. An endpoint that sends packets near the end of the idle timeout period risks having those packets discarded if its peer enters the draining state before the packets arrive. If a peer could time out within a Probe Timeout (PTO; see Section 6.6 of [QUIC-RECOVERY]), it is advisable to test for liveness
--	---

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
 U.S. Patent No. 10,742,774

	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.” <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/id/draft-ietf-quic-transport-27.txt
during the time period, no packet is received from the second node in a data stream of the non-TCP connection to keep the non-TCP connection active; and	Google infringes claim 42 and, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform that performs a method wherein, during the time period, no packet is received from the second node (e.g., the other one of the QUIC-compliant server or client, etc.) in a data stream of the non-TCP connection (e.g., QUIC connection, etc.) to keep the non-TCP connection active. See excerpt(s) below, for example (emphasis added, if any): “19.2. PING Frame <u>Endpoints can use PING frames (type=0x01) to verify that their peers are still alive or to check reachability to the peer.</u> 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. <u>A connection will time out if no packets are sent or received for a period longer than the time specified in the idle_timeout transport parameter (see Section 10) .</u> 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.” <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/html/draft-ietf-quic-transport-22

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

the non-TCP packet is not a synchronize (SYN) packet.	Google infringes claim 42 and, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform that performs a method wherein the non-TCP packet (e.g., QUIC negotiation packet, etc.) is not a synchronize (SYN) packet. See excerpt(s) below, for example (emphasis added, if any): Note: As set forth below, synchronization is required between the two sides prior to establishing a TCP-variant connection (e.g., a QUIC connection, etc.). 7.2. Version Negotiation QUIC's connection establishment begins with version negotiation, since all communication between the endpoints, including packet and frame formats, relies on the two endpoints agreeing on a version. A QUIC connection begins with a client sending a Client Initial packet (Section 5.4.1). The details of the handshake mechanisms are described in Section 7.3, but all of the initial packets sent from the client to the server MUST use the long header format - which includes the version of the protocol being used - and they MUST be padded to at least 1200 octets. 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 CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
 U.S. Patent No. 10,742,774

Claim 44 Elements	Applicability
The method of claim 43 wherein: the timeout attribute associated with the non-TCP connection is capable of being used to unilaterally cause the non-TCP connection to at least partially close, based on the timeout attribute and at least one of a formula, an expression, a function, or a policy;	Google infringes claim 43 and, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform that performs a method wherein: the timeout attribute associated with the non-TCP connection (e.g., the QUIC connection, etc.) is capable of being used to unilaterally cause the non-TCP connection to at least partially close, based on the timeout attribute and at least one of a formula, an expression, a function, or a policy (e.g., based on the idle_timeout duration, etc.). See excerpt(s) below, for example (emphasis added, if any): Note: Based upon information and belief, the “second node” term is infringed by at least a server, and the “first node” term is infringed by at least a client. 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 determined based on the value received in the idle_timeout field of the connection negotiation packet. “10.2. Idle Timeout If the idle timeout is enabled by either peer, a connection is silently closed and its state is discarded when it remains idle for longer than the minimum of the max_idle_timeouts (see Section 18.2) and three times the current Probe Timeout (PTO). <u>Each endpoint advertises a max_idle_timeout, but the effective value at an endpoint is computed as the minimum of the two advertised values.</u> By announcing a max_idle_timeout, an endpoint commits to initiating an immediate close (Section 10.3) if it abandons the connection prior to the effective value. An endpoint restarts its idle timer when a packet from its peer is received and processed successfully. The idle timer is also restarted when sending 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

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	initiating new activity. An endpoint might need to send packets to avoid an idle timeout if it is unable to send application data due to being blocked on flow control limits; see Section 4. <u>An endpoint that sends packets near the end of the idle timeout period risks having those packets discarded if its peer enters the draining state before the packets arrive.</u> If a peer could time out within a Probe Timeout (PTO; see Section 6.6 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.” <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/id/draft-ietf-quic-transport-27.txt “19.2. PING Frame Endpoints <u>can use</u> 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. <u>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).</u> 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. <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/id/draft-ietf-quic-transport-27.txt
--	--

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
 U.S. Patent No. 10,742,774

	Note: The following evidence is related to and/or describes the QUIC standard contained herein. "5. Connection Closure Once established, an HTTP/3 connection can be used for many requests and responses over time until the connection is closed. <u>Connection closure can happen in any of several different ways.</u> 5.1. Idle Connections <u>Each QUIC endpoint declares an idle timeout during the handshake. If the connection remains idle (no packets received) for longer than this duration, the peer will assume that the connection has been closed.</u> HTTP/3 implementations will need to open a new connection for new requests if the existing connection has been idle for longer than the server's advertised idle timeout, and SHOULD do so if approaching the idle timeout. HTTP clients are expected to request that the transport keep connections open while there are responses outstanding for requests or server pushes, as described in Section 10.2.2 of [QUIC-TRANSPORT]. <u>If the client is not expecting a response from the server, allowing an idle connection to time out is preferred over expending effort maintaining a connection that might not be needed.</u> A gateway MAY maintain connections in anticipation of need rather than incur the latency cost of connection establishment to servers. <u>Servers SHOULD NOT actively keep connections open.</u>" https://www.ietf.org/id/draft-ietf-quic-applicability-07.txt
the time period starts in response to the second node sending a keep-alive packet in the non-TCP connection for routing towards the first node; and	Google infringes claim 43 and, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform that performs a method wherein the time period starts in response to the second node (e.g., the other one of the QUIC-compliant server or client, etc.) sending a keep-alive packet in the non-TCP connection (e.g., QUIC connection, etc.) for routing towards the first node (e.g., one of the QUIC-compliant server or client, etc.). See excerpt(s) below, for example (emphasis added, if any): Note: As set forth below, a QUIC connection establishment begins by sending a handshake packet. "10.2. Idle Timeout

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
 U.S. Patent No. 10,742,774

	... <u>An endpoint restarts its idle timer when a packet from its peer is received and processed successfully. The idle timer is also restarted when sending an ack-eliciting packet (see [QUIC-RECOVERY]), but only if no other ack-eliciting packets have been sent since last receiving a packet.</u> Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. An endpoint might need to send packets to avoid an idle timeout if it is unable to send application data due to being blocked on flow control limits; see Section 4.” <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/id/draft-ietf-quic-transport-27.txt “19.2. PING Frame Endpoints <u>can use</u> 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. <u>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.</u> 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. <u>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</u> (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. <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/id/draft-ietf-quic-transport-27.txt
during the time period, another keep-alive packet is sent in the non-TCP	Google infringes claim 43 and, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform that performs a method wherein, during the

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
 U.S. Patent No. 10,742,774

connection for routing towards the first node, without restarting the time period.	time period, another keep-alive packet is sent in the non-TCP connection (e.g., the QUIC connection, etc.) for routing towards the first node (e.g., one of the QUIC-compliant server or client, etc.), without restarting the time period. See excerpt(s) below, for example (emphasis added, if any): “10.2. Idle Timeout . . . <u>An endpoint restarts its idle timer when a packet from its peer is received and processed successfully. The idle timer is also restarted when sending an ack-eliciting packet (see [QUIC-RECOVERY]), but only if no other ack-eliciting packets have been sent since last receiving a packet.</u> Restarting when sending packets ensures that connections do not prematurely time out when initiating new activity. An endpoint might need to send packets to avoid an idle timeout if it is unable to send application data due to being blocked on flow control limits; see Section 4.” <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/id/draft-ietf-quic-transport-27.txt “19.2. PING Frame <u>Endpoints can use PING frames (type=0x01) to verify that their peers are still alive or to check reachability to the peer.</u> The PING frame contains no additional fields. The receiver of a PING frame simply needs to acknowledge the packet containing this frame. <u>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.</u> 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 specified in the 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
---	--

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

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

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
 U.S. Patent No. 10,742,774

Claim 51 Elements	Applicability
The method of claim 43 wherein the timeout attribute associated with the non-TCP connection is used by the first node to unilaterally at least partially close the non-TCP connection.	Google infringes claim 43 and, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform that performs a method wherein the timeout attribute associated with the non-TCP connection is used by the first node (e.g., one of the QUIC-compliant server or client, etc.) to unilaterally at least partially close the non-TCP connection (e.g., the QUIC connection, etc.). See excerpt(s) below, for example (emphasis added, if any): Note: As set forth below, after the idle time period is detected, the QUIC connection is closed due to inactivity. “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).” “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.” <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> 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.

March 17, 2021

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	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 specified in the 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.” <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/html/draft-ietf-quic-transport-22 Note: The following evidence is related to and/or describes the QUIC standard contained herein. "5. Connection Closure Once established, an HTTP/3 connection can be used for many requests and responses over time until the connection is closed. <u>Connection closure can happen in any of several different ways.</u> 5.1. Idle Connections <u>Each QUIC endpoint declares an idle timeout during the handshake. If the connection remains idle (no packets received) for longer than this duration, the peer will assume that the connection has been closed.</u> HTTP/3 implementations will need to open a new connection for new requests if the existing connection has been idle for longer than the server's advertised idle timeout, and SHOULD do so if approaching the idle timeout. HTTP clients are expected to request that the transport keep connections open while there are responses outstanding for requests or server pushes, as described in Section 10.2.2 of [QUIC-
--	--

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	TRANSPORT]. <u>If the client is not expecting a response from the server, allowing an idle connection to time out is preferred over expending effort maintaining a connection that might not be needed.</u> A gateway MAY maintain connections in anticipation of need rather than incur the latency cost of connection establishment to servers. <u>Servers SHOULD NOT actively keep connections open.</u>" https://www.ietf.org/id/draft-ietf-quic-applicability-07.txt
--	--

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
 U.S. Patent No. 10,742,774

Claim 59 Elements	Applicability
The method of claim 43 wherein a user timeout attribute, that is separate from the timeout attribute, is utilized for monitoring communication of one or more acknowledgment packets, where the user timeout attribute is determined based on the time period.	Google infringes claim 43 and, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform that performs a method wherein a user timeout attribute, that is separate from the timeout attribute, is utilized for monitoring communication of one or more acknowledgment packets, where the user timeout attribute is determined based on the time period. Note: Based upon information and belief, the user timeout in QUIC is based on the max_ack_delay transport parameter negotiated during setup. It is utilized to determine a probe timeout (PTO). Note that the max_ack_delay parameter is negotiated during setup. See QUIC transport draft section 13.2.1 at Hypertext Transfer Protocol Version 3 (HTTP/3) (ietf.org), section 4.7 in the IETF QUIC recovery draft at https://tools.ietf.org/html/draft-ietf-quic-recovery-34#section-4.7, and section 10.1 in the IETF QUIC transport draft at https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-34#section-10.1. Note: Based upon information and belief, the user timeout attribute in QUIC is based on the max_ack_delay transport parameter. See section 4.7 in the IETF QUIC recovery draft at https://tools.ietf.org/html/draft-ietf-quic-recovery-34#section-4.7. The definition of max_ack_delay in the QUIC transport establishes that the user timeout is used to control ACK packets (copied below). See also section 13.2.1 "Sending ACK Frames" in the IETF QUIC transport draft at https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-34#section-13.2.1. "max_ack_delay (0x0b): 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 2¹⁴ or greater are invalid." https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-27#section-18.2 "13.2.1. Sending ACK Frames Every packet SHOULD be acknowledged at least once, and ack-eliciting packets MUST be acknowledged at least once within the maximum ack delay. An endpoint communicates its

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	maximum delay using the max_ack_delay transport parameter; see Section 18.2. max_ack_delay declares an explicit contract: an endpoint promises to never intentionally delay acknowledgments of an ack-eliciting packet by more than the indicated value. If it does, any excess accrues to the RTT estimate and could result in spurious or delayed retransmissions from the peer. For Initial and Handshake packets, a max_ack_delay of 0 is used. The sender uses the receiver's "max_ack_delay" value in determining timeouts for timer-based retransmission, as detailed in Section 5.2.1 of [QUIC-RECOVERY]." https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-27 Note: The following evidence is related to and/or describes the QUIC standard contained herein. "Case 2: TCP receives a FIN from the network If an unsolicited FIN arrives from the network, the receiving TCP can ACK it and tell the user that the connection is closing. The user will respond with a CLOSE, upon which the TCP can send a FIN to the other TCP after sending any remaining data. <u>The TCP then waits until its own FIN is acknowledged whereupon it deletes the connection. If an ACK is not forthcoming, after the user timeout the connection is aborted and the user is told.</u> ... 3.9. <u>Event Processing</u> The processing depicted in this section is an example of one possible implementation. Other implementations may have slightly different processing sequences, but they should differ from those in this section only in detail, not in substance. <u>The activity of the TCP can be characterized as responding to events.</u> The events that occur can be cast into three categories: user calls, arriving segments, and <u>timeouts</u>. This section describes the processing the TCP does in response to each of the events. In many cases the processing required depends on the state of the connection. ... Timeouts <u>USER TIMEOUT</u> <u>RETRANSMISSION TIMEOUT</u>
--	---

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	TIME-WAIT TIMEOUT ... OPEN Call CLOSED STATE (i.e., TCB does not exist) <u>Create a new transmission control block (TCB) to hold connection state information. Fill in local socket identifier, foreign socket, precedence, security/compartment, and user timeout information.</u> Note that some parts of the foreign socket may be unspecified in a passive OPEN and are to be filled in by the parameters of the incoming SYN segment. Verify the security and precedence requested are allowed for this user, if not return "error: precedence not allowed" or "error: security/compartment not allowed." If passive enter the LISTEN state and return. If active and the foreign socket is unspecified, return "error: foreign socket unspecified"; if active and the foreign socket is specified, issue a SYN segment. An initial send sequence number (ISS) is selected. A SYN segment of the form <SEQ=ISS><CTL=SYN> is sent. Set SND.UNA to ISS, SND.NXT to ISS+1, enter SYN-SENT state, and return." https://tools.ietf.org/html/rfc793 (emphasis added)
--	--

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
 U.S. Patent No. 10,742,774

Claim 65 Elements	Applicability
The method of claim 43 wherein the timeout attribute associated with the non-TCP connection is capable of being used by the first node to unilaterally at least partially close the non-TCP connection, without sending to the second node any packet that is related to the detection of the time period.	Google infringes claim 43 and, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform that performs a method wherein the timeout attribute associated with the non-TCP connection (e.g., the QUIC connection, etc.) is capable of being used by the first node (e.g., one of the QUIC-compliant server or client, etc.) to unilaterally at least partially close the non-TCP connection, without sending to the second node (e.g., the other one of the QUIC-compliant server or client, etc.) any packet that is related to the detection of the time period. See excerpt(s) below, for example (emphasis added, if any): Note: As set forth below, after the idle time period is detected, the QUIC connection is closed due to inactivity. “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).” “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.” <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> 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.

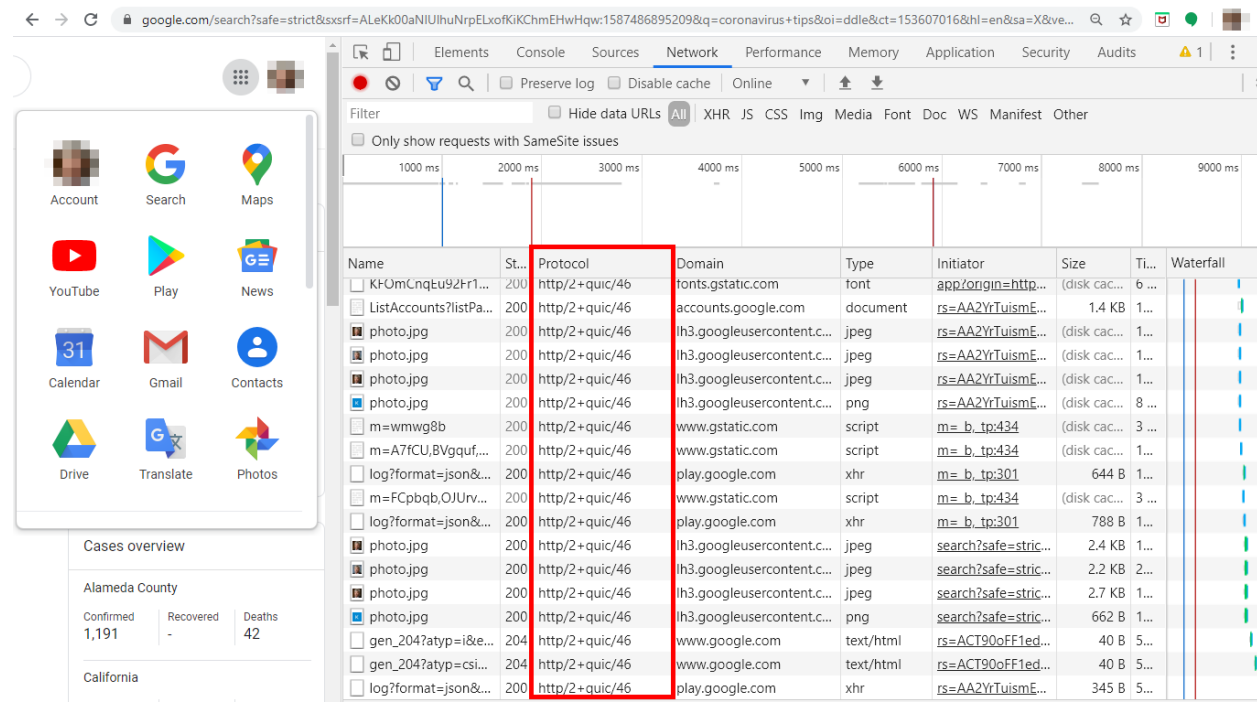
CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	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 specified in the 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.” <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/html/draft-ietf-quic-transport-22 Note: The following evidence is related to and/or describes the QUIC standard contained herein. "5. Connection Closure Once established, an HTTP/3 connection can be used for many requests and responses over time until the connection is closed. <u>Connection closure can happen in any of several different ways.</u> 5.1. Idle Connections <u>Each QUIC endpoint declares an idle timeout during the handshake. If the connection remains idle (no packets received) for longer than this duration, the peer will assume that the connection has been closed.</u> HTTP/3 implementations will need to open a new connection for new requests if the existing connection has been idle for longer than the server's advertised idle timeout, and SHOULD do so if approaching the idle timeout.
--	---

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	HTTP clients are expected to request that the transport keep connections open while there are responses outstanding for requests or server pushes, as described in Section 10.2.2 of [QUIC-TRANSPORT]. <u>If the client is not expecting a response from the server, allowing an idle connection to time out is preferred over expending effort maintaining a connection that might not be needed.</u> A gateway MAY maintain connections in anticipation of need rather than incur the latency cost of connection establishment to servers. <u>Servers SHOULD NOT actively keep connections open.</u>" https://www.ietf.org/id/draft-ietf-quic-applicability-07.txt
--	--

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

Claim 67 Elements	Applicability
The method of claim 43 wherein the non-TCP connection is not a TCP extension.	Google infringes claim 43 and, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform that performs a method wherein the non-TCP connection (e.g., QUIC connection, etc.) is not a TCP extension. See excerpt(s) below, for example (emphasis added, if any): Note: Below is a web page of Google (https://www.google.com/).  Note: As set forth below, QUIC is a “variant” of TCP.

March 17, 2021

77 of 81

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	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 Note: The following evidence is related to and/or describes the QUIC standard contained herein. “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
--	--

CLAIM CHARTS
 BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
 U.S. Patent No. 10,742,774

Claim 72 Elements	Applicability
The method of claim 43 wherein the time period is not a user timeout period.	Google infringes claim 43 and, through its products/services owns, controls, and/or uses at least one server computer and/or the Google Cloud Platform that performs a method wherein the time period is not a user timeout period. See excerpt(s) below, for example (emphasis added, if any): Note: As set forth below, the metadata includes a value (time period) in seconds that is encoded as an unsigned 16-bit integer, from which the user timeout period is separate. idle_timeout (0x0003): The idle timeout is a value in seconds that is encoded as an unsigned 16-bit integer. The maximum value is 600 seconds (10 minutes). 7.8.2. Idle Timeout A connection that remains idle for longer than the idle timeout (see Section 7.4.1) becomes closed. Either peer removes connection state if they have neither sent nor received a packet for this time. The time at which an idle timeout takes effect won't be perfectly synchronized on peers. A connection enters the draining period when the idle timeout expires. During this time, an endpoint that receives new packets MAY choose to restore the connection. Alternatively, an endpoint that receives packets MAY signal the timeout using an immediate close. <i>QUIC: A UDP-Based Multiplexed and Secure Transport</i> https://tools.ietf.org/html/draft-ietf-quic-transport-07 Note: The following evidence is related to and/or describes the QUIC standard contained herein. “3.9. Event Processing

March 17, 2021

79 of 81

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

	The processing depicted in this section is an example of one possible implementation. Other implementations may have slightly different processing sequences, but they should differ from those in this section only in detail, not in substance. <u>The activity of the TCP can be characterized as responding to events.</u> The events that occur can be cast into three categories: user calls, arriving segments, and <u>timeouts</u>. This section describes the processing the TCP does in response to each of the events. In many cases the processing required depends on the state of the connection. ... Timeouts <u>USER TIMEOUT</u> RETRANSMISSION TIMEOUT TIME-WAIT TIMEOUT ... OPEN Call CLOSED STATE (i.e., TCB does not exist) <u>Create a new transmission control block (TCB) to hold connection state information. Fill in local socket identifier, foreign socket, precedence, security/compartments, and user timeout information.</u> Note that some parts of the foreign socket may be unspecified in a passive OPEN and are to be filled in by the parameters of the incoming SYN segment. Verify the security and precedence requested are allowed for this user, if not return "error: precedence not allowed" or "error: security/compartments not allowed." If passive enter the LISTEN state and return. If active and the foreign socket is unspecified, return "error: foreign socket unspecified"; if active and the foreign socket is specified, issue a SYN segment. An initial send sequence number (ISS) is selected. A SYN segment of the form <SEQ=ISS><CTL=SYN> is sent. Set SND.UNA to ISS, SND.NXT to ISS+1, enter SYN-SENT state, and return." https://tools.ietf.org/html/rfc793 (emphasis added)
--	--

CLAIM CHARTS
BASED ON INFRINGEMENT ANALYSIS OF GOOGLE
U.S. Patent No. 10,742,774

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.