IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
MARSHALL DIVISION

ADVANCED CODING TECHNOLOGIES LLC,	§	Case No. 2:24-cv-00353-JRG
	§	**JURY TRIAL DEMANDED**
Plaintiff,	§	
v.	§	
GOOGLE LLC,	§	
Defendant.	§	

FIRST AMENDED COMPLAINT FOR PATENT INFRINGEMENT

Plaintiff Advanced Coding Technologies LLC ("ACT" or "Plaintiff") for its Complaint against Defendant Google LLC ("Google" or "Defendant") for patent infringement under 35 U.S.C. § 271, alleges as follows:

THE PARTIES

1. Plaintiff ACT is a limited liability company, organized and existing under the laws of the State of Texas, with its principal place of business located at 104 East Houston Street, Suite 140, Marshall, Texas 75670.

2. Defendant Google is a Delaware corporation and maintains its principal place of business at 1600 Amphitheatre Parkway, Mountain View, California 94043, and may be served with process via its registered agent, Corporation Service Company at 251 Little Falls Drive, Wilmington, Delaware 19808. Upon information and belief, Google does business in Texas, directly or through intermediaries, and offers its products and/or services, including those accused herein of infringement, to customers and potential customers located in Texas, including in the

APPLE 1073
Apple v. Advanced Coding Techn.
IPR2025-01221

judicial Eastern District of Texas.

JURISDICTION

3. This is an action for patent infringement arising under the patent laws of the United States, 35 U.S.C. §§ 1, *et seq*. This Court has jurisdiction over this action pursuant to 28 U.S.C. §§ 1331, 1332, 1338 and 1367.

4. This Court has specific and personal jurisdiction over Defendant consistent with the requirements of the Due Process Clause of the United States Constitution and the Texas Long Arm Statute. Upon information and belief, the Defendant has sufficient minimum contacts with the forum because Defendant transacts substantial business in the State of Texas and in this Judicial District. Further, Defendant has, directly or through subsidiaries or intermediaries, committed and continues to commit acts of patent infringement in the State of Texas and in this Judicial District as alleged in this Complaint, by, among other things, offering to sell and selling products and/or services that infringe the Patents-in-Suit.

5. Venue is proper in this Judicial District pursuant to 28 U.S.C. §§ 1391 and 1400(b). Google is registered to do business in Texas and, upon information and belief, Google has transacted business in the Eastern District of Texas and has committed acts of direct and indirect infringement in the Eastern District of Texas. Google has regular and established places of business in this Judicial District as set forth below and is deemed to reside in this Judicial District.

6. Google is a multi-national technology company that collects, stores, organizes, and distributes data. In addition to its service model for distribution of data (*e.g.*, movies, search results, maps, music, etc.), Google has an expansive regime that gathers data on residents of this District through the hardware devices it sells (*e.g.*, phones, tablets, and home audio devices) and, also, through the operating systems and apps it provides. As an example, Google gathers data when a

resident runs its operating systems and apps (*e.g.,* location services).[1] As another example, Google gathers data when a resident interacts with Google's plethora of services such as search, email, music, and movie streaming. *See* https://safety.google/privacy/data/ (indicating that Google gathers data from "things you search for," "Videos you watch," "Ads you view or click," "Your location," "Websites you visit," and "Apps, browsers, and devices you use to access Google services"). As yet another example, Google gathers data from "where you've been," "everything you've ever searched—and deleted," "all the apps you use," "all of your YouTube history," "which events you attended, and when," "information you deleted [on your computer]," "your workout routine," "years' worth of photos," and "every email you ever sent."[2] In addition to extensive data gathering of information on residents of this District, Google has a substantial presence in this District directly through the products and services Google provides residents of this District (some of which also gather data).[3]

7. Google describes itself as an "information company."[4] Its vision is "to provide access to the world's information in one click," and its mission is "to organize the world's information and make it universally accessible and useful."[5] Making information available to

---

[1] *See e.g.*, "AP Exclusive: Google tracks your movements, like it or not," https://apnews.com/828aefab64d4411bac257a07c1af0ecb/AP-Exclusive:-Google-tracks-yourmovements,-like-it-or-not

[2] *See* https://www.theguardian.com/commentisfree/2018/mar/28/all-the-data-facebook-google-has-on-you-privacy

[3] Non-limiting examples include Google Search, Maps, Translate, Chrome Browser, YouTube, YouTube TV, Google Play Music, Chromecast, Google Play Movies and TV, Android Phones, Android Wear, Chromebooks, Android Auto, Gmail, Google Allo, Google Duo, Google+, Google Photos, Google Contacts, Google Calendar, Google Keep, Google Docs, Google Sheets, Google Slides, Google Drive, Google Voice, Google Assistant, Android operating system, Project Fi Wireless phone systems, Google Pixel, Google Home, Google Wifi, Daydream View, Chromecast Ultra.

[4] *See* "This Year's Founder's Letter" by Alphabet CEO, Sundar Pichai, https://blog.google/inside-google/alphabet/this-years-founders-letter//

[5] https://panmore.com/google-vision-statement-mission-statement

people wherever they are and as quickly as possible is critical to Google's business.

Google Global Cache (GGC)

8. Google's CEO, Sundar Pichai, explained, "We want to make sure that no matter who you are and where you are or how advanced the device you are using—Google works for you."[6] To meet this goal, Google developed a content delivery network that it calls the Edge Network.

9. One non-limiting example of physical presence in this Judicial District is Google's Edge Network. Google provides web-based products and services, such as Google Maps, Find My Device, and Google Chrome, to users throughout the world, including in this Judicial District. These products and services are in high demand. Google reports that the Android operating system has more than 2 billion monthly active devices, and Google Maps surpassed 1 billion users as of May 2017.[7]

10. Google's Edge Network, itself, has three elements: Core Data Centers, Edge Points of Presence, and Edge Nodes.[8] The Core Data Centers (there are eight in the United States) are used for computation and backend storage. Edge Points of Presence are the middle tier of the Edge Network and connect the Data Centers to the internet. Edge Nodes are the layer of the network closest to users. Popular content, including Google Maps, Google Messages, mobile apps, and other digital content from the Google Play store, is cached on the Edge Nodes, which Google refers to as Google Global Cache or "GGC."

---

[6] https://time.com/4311233/google-ceo-sundar-pichai-letter/
[7] *See* https://www.theverge.com/2017/5/17/1564333/android-reaches-2-billion-monthly-active-users
[8] https://peering.google.com/#/infrastructure

11. Google Global Cache is recognized as one of Google's most important pieces of infrastructure,"[9] and Google uses it to conduct the business of providing access to the world's information. GGC servers in the Edge Nodes function as local data warehouses, much like a shoe manufacturer might have warehouses around the country. Instead of requiring people to obtain information from distant Core Data Centers, which would introduce delay, Google stores information in the local GGC servers to provide quick access to the data.

12. Caching and localization are vital for Google's optimization of network resources. Because hosting all content everywhere is inefficient, it makes sense to cache popular content and serve it locally. Doing so brings delivery costs down for Google, network operators, and internet service providers. Storing content locally also allows it to be delivered more quickly, which improves user experience. Serving content from the edge of the network closer to the user improves performance and user happiness. To achieve these benefits, Google has placed Edge Nodes throughout the United States, including in this Judicial District. Google describes these Edge Nodes as the workhorses of video delivery.

13. Google's GGC servers are housed in spaces in this Judicial District leased by Google. Google's GGC servers are housed in spaces leased by Google from Internet Service Providers (ISPs) whose networks have substantial traffic to Google and are interested in saving bandwidth. Hosting Google servers allows ISPs to save both bandwidth and costs, as they do not incur the expense of carrying traffic across their peering and/or transit links.

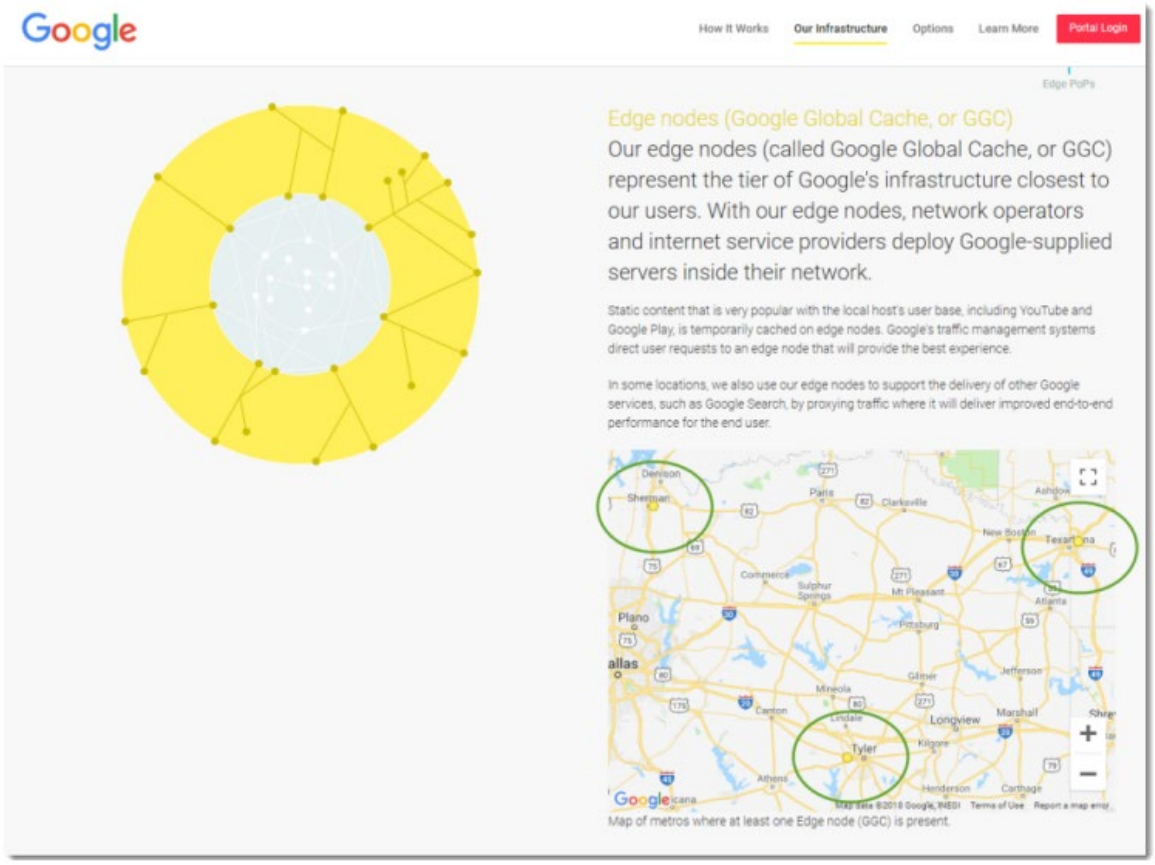
14. When an ISP agrees to host a GGC server, the parties enter into a Global Cache Service Agreement, under which Google provides:

[9] https://www.blog.speedchecker.xyz/2015/11/30/demystifying-google-global-cache/

- hardware and software—including GGC servers and software—to be housed in the host's facilities;
- technical support; service management of the hardware and software; and
- content distribution services, including content caching and video streaming.

In exchange, the host provides, among other things, a physical building, rack space where Google's computer hardware is mounted, power, and network interfaces. All ownership rights, title, and intellectual property rights in and to the equipment (*i.e.*, the hardware and software provided by Google) remain with Google and/or its licensors.

15. Multiple ISP-hosted GGC servers are in this Judicia District. Google provides the location of its GGC servers, namely, Sherman, Tyler, and Texarkana.



Source: Uniloc 2017 LLC v. Google LLC, Case No. 2:18-cv-00550, Dkt. 1 at 8 (E.D. Tex. 2018); https://peering.google.com/#/infrastructure.

16. Suddenlink Communications, for example, is an ISP that hosts six GGC servers in Tyler, Texas.

17. CableOne is an ISP that hosts three GGC servers in Sherman, Texas and three GGC servers in Texarkana, Texas.

18. Google caches content on these GGC servers located in this Judicial District.

19. Google's GGC servers located in this Judicial District cache content that includes, among other things: (a) maps; (b) messages; and (c) digital content from the Google Play store.

20. Google's GGC servers located in this Judicial District deliver cached content for the items in the preceding paragraph to residents in this Judicial District.

21. Google generates revenue (a) by delivering video advertising; (b) from apps; and (c) from digital content in the Google Play store.

22. Google treats its GGC servers in this Judicial District the same as it treats all of its other GGC servers in the United States.

23. The photographs below show Google's GGC servers hosted by Suddenlink and the building where they are located at 322 North Glenwood Boulevard, Tyler, Texas 75702.

Exterior

Interior Rack Spaces

Google GGC Servers

24. Google not only exercises exclusive control over the digital aspects of the GGC, but also exercises exclusive control over the physical server and the physical space within which

the server is located and maintained.

25. This Judicial District has previously determined that the GGC server itself and the place of the GGC server, both independently and together, meet the statutory requirement of a "physical place." *See Seven Networks, LLC v. Google LLC*, Case No. 2:17-cv-00442-JRG, Dkt. 235 at 24 (E.D. Tex. July 19, 2018).

26. Likewise, this Judicial District has determined that GGC servers and their several locations within this Judicial District constitute "regular and established place[s] of business" within the meaning of the special patent venue statute. *See Seven Networks, LLC v. Google LLC*, Case No. 2:17-cv-00442-JRG, Dkt. 235 at 38 (E.D. Tex. July 19, 2018).

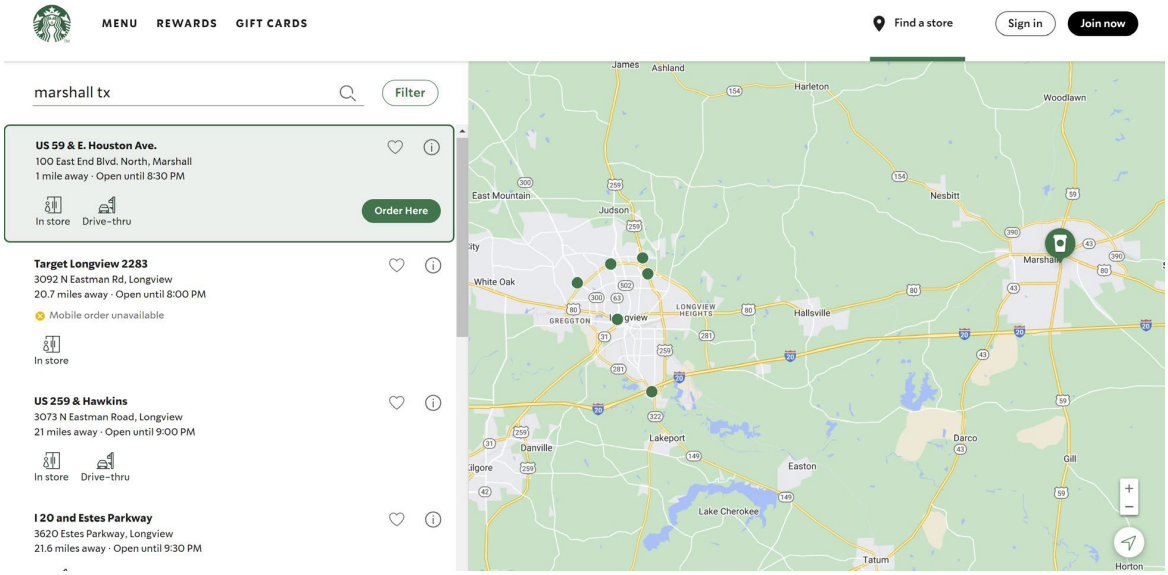
27. Similarly, this Judicial District has determined that the GGC servers and their locations within the various ISPs within this Judicial District are "places of Google" sufficient to meet the statutory requirement of § 1400(b). *See Seven Networks, LLC v. Google LLC*, Case No. 2:17-cv-00442- JRG, Dkt. 235 at 41 (E.D. Tex. July 19, 2018).

Google's Google Wi-Fi at Starbucks Locations in this Judicial District

28. Google provides Wi-Fi infrastructure and Wi-Fi service at Starbucks locations in this Judicial District. Google and Starbucks entered into an agreement in which Google provides its Google Wi-Fi or Google Fiber service at all Starbucks locations in this Judicial District, including at Starbucks stores and at Target stores.[10] First-time customers connect and use Google Wi-Fi on their devices in this Judicial District by selecting "Google Starbucks" from their respective device's list of available wireless networks and entering their respective name, email address, and postal code. Return customers are automatically connected to Google Wi-Fi on their respective devices at any Google Wi-Fi location. Upon connecting to the Google Wi-Fi locations

[10] https://www.starbucks.com/store-locator?map=32.467135,-95.387478,8z

in this Judicial District, Google provides connected customers with Internet access over Google's infrastructure and services.



Source: https://www.starbucks.com/store-locator?map=32.49512,-94.568225,11z&place=marshall%20tx

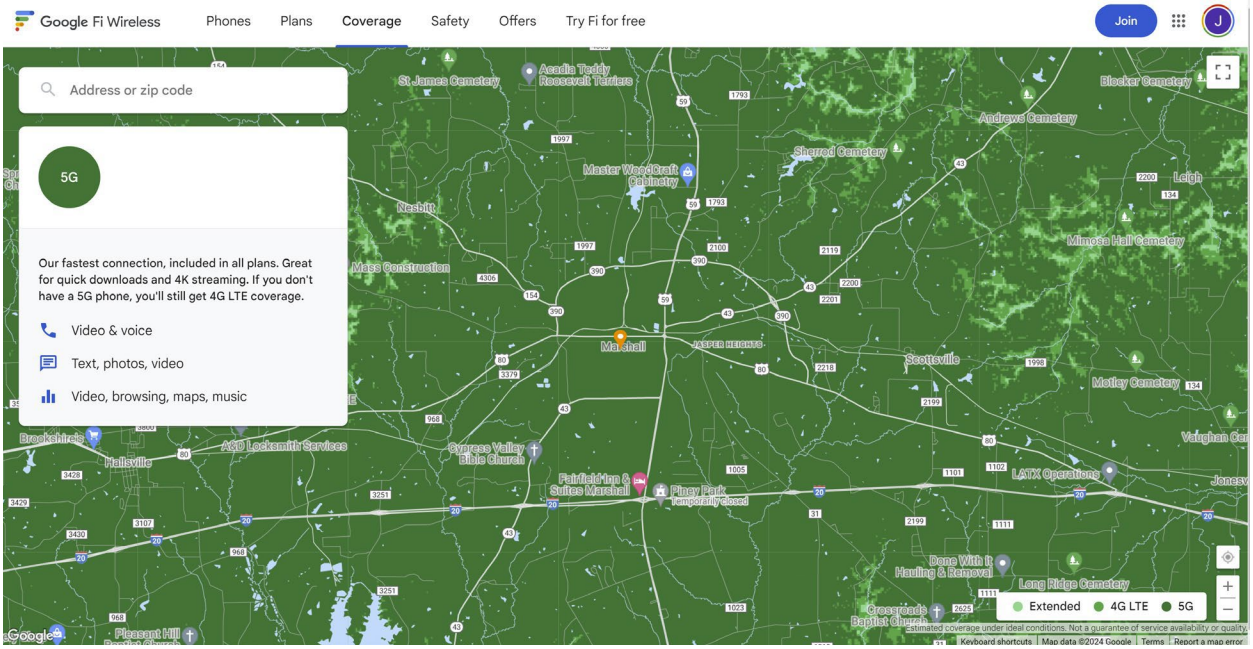
29. Google uses its Google Wi-Fi infrastructure and Google Wi-Fi services at Starbucks locations in this Judicial District to provide customers with telecommunications services through its own phone carrier network, Google Fi. Google Fi is owned and operated by Google. In order to use Google Fi phone service in this Judicial District, Google provides its customers with special SIM cards and software to connect to and automatically switch between four sources of network infrastructure and services: T-Mobile, Sprint, US Cellular, and public Wi-Fi networks. As described below, Google has entered into agreements with T-Mobile, Sprint, and US Cellular to lease the carriers' infrastructure and services to provide Google Fi customers with voice and data services. As a fourth source, Google Fi uses public Wi-Fi networks, including the Google Wi-Fi at Starbucks locations in this Judicial District, to provide its phone carrier service. The Google Wi-Fi at Starbucks locations in this Judicial District are fixed geographical locations. They are "regular" and "established" because they operate in a "steady, uniform, orderly, and methodical

manner" and are sufficiently permanent. They are "of the defendant" because Google has contractual and/or property rights to use the Google Wi-Fi locations to operate its businesses, including the Google Fi phone carrier business.

30. Google determines whether a Google Fi customer in this Judicial District uses a certain Wi-Fi network, including the Google Wi-Fi networks at Starbucks locations, using the Google-provided SIM card and software on the customer's phone.

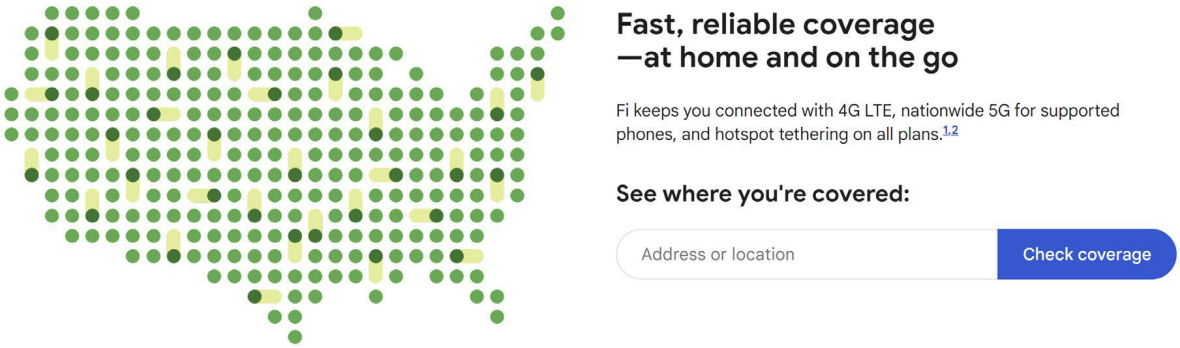
Google's "Google Fi"

31. As described above, Google owns, operates, and provides telecommunications infrastructure and service in this Judicial District through its own phone carrier network, Google Fi. Google provides cellular and Wi-Fi infrastructure and services for phone, messaging, and data services in this Judicial District. Google provides its customers voice and high-speed data coverage (4G LTE) for cities such as Tyler, Longview, and Marshall, Texas.



Source: https://fi.google.com/coverage?q=marshall%20tx

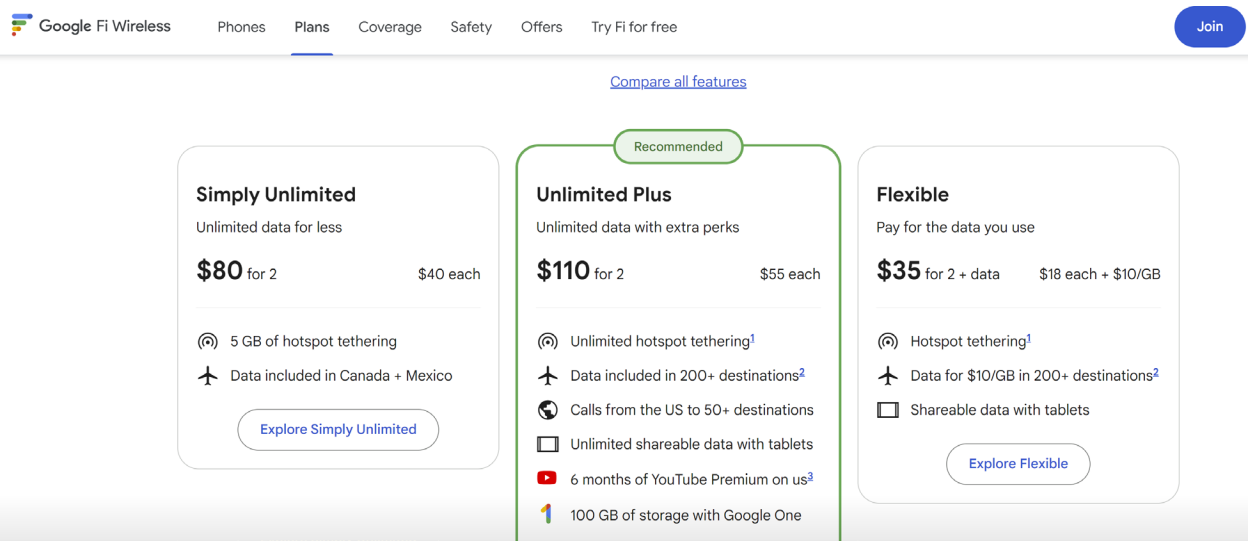
32. The cell towers used for Google's services are fixed geographical locations. They are "regular" and "established" because they operate in a "steady, uniform, orderly, and methodical manner" and are sufficiently permanent. They are "of the defendant" because Google has contractual and/or property rights to use the cell towers to operate its business. Google also ratifies the service locations through its coverage lookup service.



Source: https://fi.google.com/about/coverage

33. With this coverage lookup service, Google advertises its ability to provide cell coverage in this Judicial District and its selected cell towers in and near this Judicial District to provide the advertised coverage (*e.g.*, 2G, 3G, or 4G LTE) depending on the location in the Judicial District. *See* https://fi.google.com/about/coverage/. Google is not indifferent to the location of its cell towers. It "established" and "ratified" them where they are for a specific business purpose.

34. Residents of this Judicial District also directly contract with and are billed by Google for these services.



Source: https://fi.google.com/about/plan

Google Cloud Interconnect (GCI) and Direct Peering

35. Google additionally services its customers in this Judicial District (and other districts) through yet other facilities it has in this Judicial District. More specifically, Google's equipment is located in this Judicial District in Denton County, Texas at two facilities referred to as "Megaport." At the Megaport facilities in this Judicial District, Google offers two services: Google Cloud Interconnect (GCI) and Direct Peering.

36. Google's Cloud Interconnect (GCI) is a service from Google that allows customers to connect to Google's Cloud Platform directly, as opposed to, for example, over the public network.

Interconnect > Documentation

Partner Interconnect Overview

☆☆☆☆☆
SEND FEEDBACK

Google Cloud Interconnect - Partner (Partner Interconnect) provides connectivity between your on-premises network and your VPC network through a supported service provider. A Partner Interconnect connection is useful if your data center is in a physical location that can't reach a Dedicated Interconnect colocation facility or if your data needs don't warrant an entire 10 Gbps connection.

Before you use Partner Interconnect

★ **Note:** Partner Interconnect requires that you separately obtain services from a third-party network service provider. Google is not responsible for any aspects of Partner Interconnect provided by the third-party service provider nor any issues outside of Google's network.

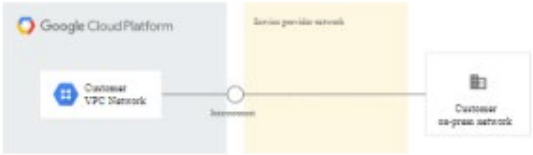
- You must be familiar with the Cloud Interconnect terminology described in Key Terminology.
- You must work with a supported service provider to establish connectivity between their network and your on-premises network.

How does Partner Interconnect work?

Service providers have existing physical connections to Google's network that they make available for their customers to use.

After you establish connectivity with a service provider, you can request a Partner Interconnect connection from your service provider. After the service provider provisions your connection, you can start passing traffic between your networks by using the service provider's network.

The following diagram provides a high-level overview of a customer using a service provider to connect to Google:



Basic Partner Interconnect topology (click to enlarge)

Source: https://cloud.google.com/interconnect/docs/concepts/partner-overview

37. Google's Direct Peering services allow its customers to exchange Internet traffic between its customers' networks and Google's at one of its broad-reaching Edge network locations, such as the one at Megaport.

Interconnect > Documentation > Google Cloud

Direct Peering

SEND FEEDBACK

Direct Peering allows you to establish a direct peering connection between your business network and Google's edge network and exchange high-throughput cloud traffic. This capability is available at any of more than 100 locations in 33 countries around the world. Visit Google's peering site to find out more information about Google's edge locations.

When established, Direct Peering provides a direct path from your on-premises network to Google services, including the full suite of Google Cloud Platform products. Traffic from Google's network to your on-premises network also takes that direct path, including traffic from VPC networks in your projects. GCP customers must request direct egress pricing be enabled for each of their projects after they have established direct peering with Google. Refer to pricing for details.

Considerations

If used with GCP, Direct Peering doesn't produce any custom routes in a VPC network. Traffic sent from resources in a VPC network leaves by way of a route whose next hop is either a *default Internet gateway* (a default route, for example) or a Cloud VPN tunnel. If the destination for the traffic matches your on-premises IP ranges, it could be eligible for discounted egress rates, as described below.

To send traffic through Direct Peering using a route whose next hop is a Cloud VPN tunnel, the IP address of your on-premises network's VPN gateway must be in your configured destination range.

Direct Peering exists outside of Google Cloud Platform. Instead of Direct Peering, the recommended methods of access to GCP are Cloud Interconnect – Dedicated or Cloud Interconnect – Partner.

See the next section to determine which of these solutions is right for you.

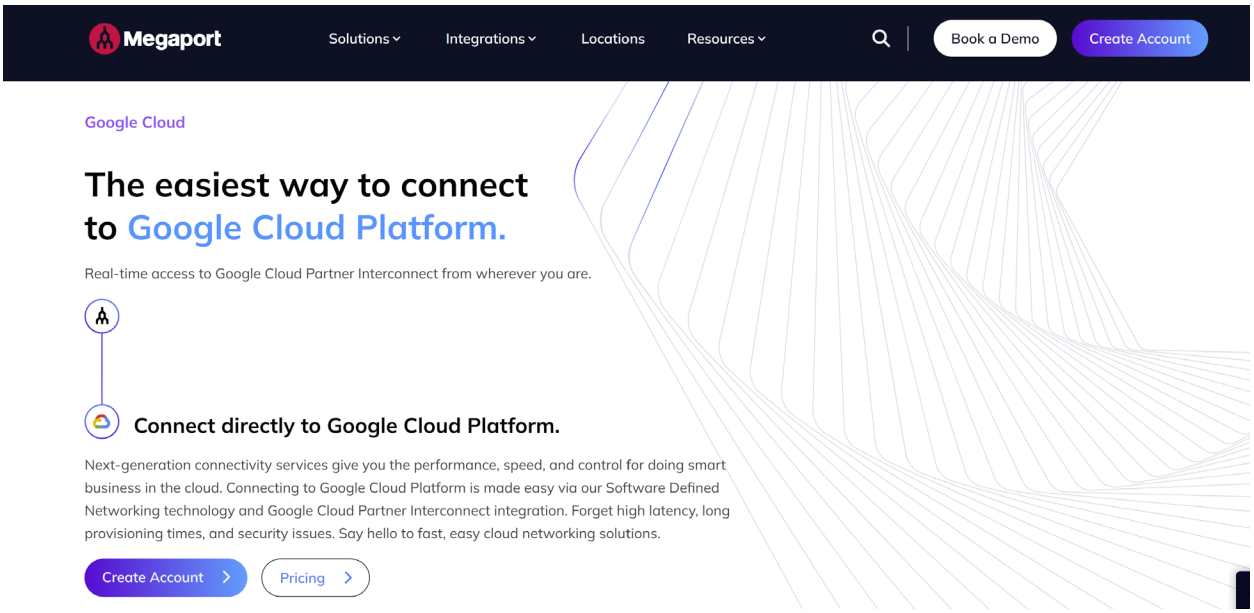
Source: https://cloud.google.com/interconnect/docs/how-to/direct-peering

38. In establishing such a direct connection, Google provides the necessary physical equipment at Megaport to enable GCI or Direct Peering connections. At least the Megaport facility shown below is located in this Judicial District and is advertised by Google as a GCI facility.

Dallas	Alestra	Layer 2 and 3
	Arelion	Layer 2 and 3
	C3NTRO	Layer 2 and 3
	Console Connect by PCCW Global	Layer 2 and 3
	Cox	Layer 2 and 3
	DE-CIX	Layer 2 and 3
	Equinix	Layer 2 and 3
	InterCloud	Layer 2 and 3
	Internet2	Layer 2 and 3
	Lumen	Layer 2 and 3
	MCM Telecom	Layer 2 and 3
	Megaport	Layer 2 and 3
	PacketFabric	Layer 2 and 3
	Transtelco	Layer 2 and 3

Source: https://cloud.google.com/network-connectivity/docs/interconnect/concepts/service-providers#north-america

39. Clicking on the Megaport link from the screenshot of Google's website in the preceding paragraph directs a customer to the details for directly connecting to Google's equipment at the facility in this Judicial District to connect to Google's GCI service.



Source: https://www.megaport.com/services/google-cloud-partner-interconnect/

40. More particularly, the Google-linked Megaport site explains how a Google customer can use the Google Cloud Platform console to enable connection to the Google equipment at the Megaport facility in this Judicial District.

How to Create a VXC to Google Cloud Platform

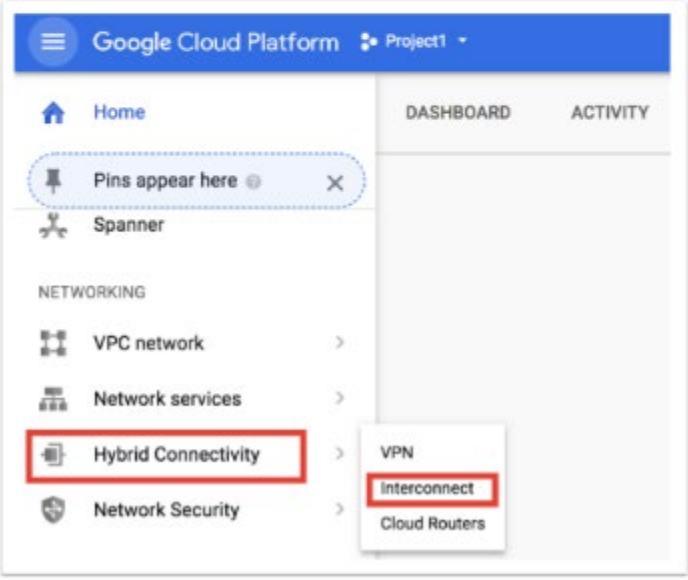
Prerequisites:

- The customer must create a Partner Interconnect attachment in Google Cloud Console or gcloud CLI.
- The Pairing Key is provided as part of the attachment creation and will need to be copied and applied in the Portal.

VXC Deployment Steps

First, you will need to log in to your Google Cloud Console and create a Pairing Key: Google Console Link

Next, click on the main menu in the Google Console, then select 'Hybrid Connectivity' and 'Interconnect' from the drop-down.



Source: Source: https://docs.megaport.com/cloud/megaport/google/

41. Both Google's website and Megaport's website advertise the peering service and point a consumer to the website, www.peeringdb.com, for details. The peering DB website lists Megaport Dallas as a Google peering facility.

Who can peer with Google?

Google recommends Google Cloud customers to use a Verified Peering Provider instead of Direct Peering.

Connecting with a Verified Peering Provider lets Google customers reach all publicly available Google resources without the complexity of managing Direct Peering connectivity to Google.

Google Cloud customers that choose a Verified Peering Provider do not need to meet Google's Direct Peering requirements and can work directly with a Verified Peering Provider to acquire internet services that provide access to Google.

To view a list of available Verified Peering Providers, see their connectivity to Google, learn about their services, and find providers in different areas, see the Google Edge Network.

Google recommends Google Cloud customers who do not use a Verified Peering Provider to privately peer with Google. Private peering provides dedicated physical ports between Google and customers, and can provide better performance and reliability than public peering.

When privately peering, Google requires physical redundancy with at least two separate connections to Google in a singl metropolitan area. Each physical connection must have its own IP addressing.

Any Google Cloud customers that meet Google's technical peering requirements can be considered for Direct Peering. Google can peer at locations listed in our PeeringDB entry.

For more information about requirements, and to review Google's peering best practices for Google Cloud customers, visit Google peering.

Source: https://cloud.google.com/interconnect/docs/how-to/direct-peering

Megaport - Google IX Peering Locations:

- MegaIX: Ashburn, Dallas, Los Angeles, Seattle, Singapore, Sofia, Sydney
- AMS-IX: Chicago, New York, Bay Area

See PeeringDB for additional details.

Source: https://knowledgebase.megaport.com/cloud-connectivity/google-cloud-platform-direct-peering/

PeeringDB

Search here for a network, IX, or facility.

Advanced Search | Legacy Search

Registeror | Login

English (English)

Google LLC Platinum Sponsor

Organization	Google LLC
Also Known As	Google, YouTube (for Google Fiber see AS16591 record)
Long Name	
Company Website	https://about.google/intl/en/
ASN	15169
IRR as-set/route-set	RADB::AS-GOOGLE
Route Server URL	
Looking Glass URL	
Network Types	Content
IPv4 Prefixes	15000
IPv6 Prefixes	10000
Traffic Levels	Not Disclosed
Traffic Ratios	Mostly Outbound
Geographic Scope	Global
Protocols Supported	Unicast IPv4 Multicast IPv6 Never via route servers

Public Peering Exchange Points

dallas

Exchange AZ IPv4	ASN IPv6	Speed	RS Peer	BFD Support
DE-CIX Dallas 206.53.202.109	15169 2001:504:61::3b41:0:2	100G		
DE-CIX Dallas 206.53.202.11	15169 2001:504:61::3b41:0:1	100G		
Digital Realty Dallas 206.126.114.20	15169 2001:504:17:114::20	10G		
Equinix Dallas 206.223.118.137	15169 2001:504:0:5:0:1:5169:1	100G		
Equinix Dallas 206.223.119.16	15169 2001:504:0:5:0:1:5169:2	100G		
MegaIX Dallas 206.53.174.30	15169 2606:a980:0:7::1e	10G		
MegaIX Dallas 206.53.174.7	15169 2606:a980:0:7::7	10G		

Source: https://www.peeringdb.com/net/433

42. Megaport's website also confirms, in its "Looking Glass" tool, the presence of Google at its facility—(AS No. 15169).

Alice
Your friendly bird looking glass.

ROUTE SERVERS

Dallas IX

Dallas IX Route Server 1 (IPv4)

Dallas IX Route Server 1 (IPv6)

Dallas IX Route Server 2 (IPv4)

Dallas IX Route Server 2 (IPv6)

Dallas IX Route Server 1 (IPv4)

Filter by Neighbor, ASN or Description

BGP SESSIONS ESTABLISHED

Neighbour	ASN	State	Uptime	Description
206.53.174.37	396422	up	4 days	3DS Communications LLC
206.53.174.16	23367	up	a year	Adaptive Data Networks, LLC
206.53.174.6	40731	up	7 months	BICENTEL LLC
206.53.174.40	30081	up	25 days	CacheNetworks, LLC
206.53.174.11	13335	up	11 days	Cloudflare Inc.
206.53.174.32	33570	up	4 months	Cloudpath
206.53.174.18	49362	up	3 months	DSV A/S
206.53.174.36	25875	up	9 months	FriendFinder Networks, Inc.
206.53.174.7	15169	up	2 months	Google inc..
206.53.174.14	6939	up	11 days	Hurricane Electric
206.53.174.31	396356	up	2 months	Latitude.sh
206.53.174.12	8075	up	10 months	Microsoft

Source: https://lg.megaport.com/

43. Both of Megaport's "Dallas" locations are in the Eastern District of Texas in Denton County.[11] The larger Megaport facility, the Carrollton facility, is located at 1649 West Frankford Road and is the largest of its kind in the State of Texas.[12] The smaller Megaport facility, the Lewisville facility, is located at 2501 S. State Highway 121.[13]

44. The Google equipment at Megaport's facilities which provides the GCI and Direct Peering services for Google customers are fixed geographical locations. They are "regular" and "established" because they operate in a "steady, uniform, orderly, and methodical manner" and are sufficiently permanent. They are "of the defendant" because Google holds contractual and/or property rights to use this space and to maintain this equipment. Google also ratifies the equipment through advertising of the Megaport locations as authorized to provide these Google services.

Other Google Presence in this Judicial District

45. In addition to the Google presence described above, Google has other pervasive contracts in this Judicial District.

46. Google has multiple authorized repair centers in the Eastern District of Texas. A resident can visit Google's website to find a list of these repair centers:

---

[11] https://www.megaport.com/megaport-enabled-locations/?locationId=102
[12] *Id.*
[13] *Id.*

Find a repair partner

Use the table below to find available repair partners in your region. If there isn't an option available for your region, contact us.

Choose your country: United States

Region	Provider	Devices	Type of service	Repair type	Contact
United States	Google	All Pixel phones	Mail-in and walk-in	In-warranty and out-of-warranty	Google *
		Pixel Tablet	Mail-in	Out-of-warranty	Google
	Asurion or uBreakiFix	All Pixel phones	Mail-in and walk-in	Out-of-warranty	**Phone**: +1 877-320-2237 Asurion or uBreakiFix
		Pixel 6 and later, including Fold	Mail-in and walk-in	In-warranty and out-of-warranty	

*For the most up-to-date partner locations and options for your specific device or damage, refer to Google .

In US and Canada, replacement of parts or product service is made available for a minimum of 3 years after end of production for all Pixel devices through Google or its service providers.

Source: https://support.google.com/store/answer/7182296?hl=en

47. Google's only authorized walk-in repair center, uBreakiFix by Asurion, lists at least four facilities in this Judicial District:

Asurion uBreakiFix MY REPAIRS ENGLISH (US) FRANCHISING

ubreakifix BY asurion

Find Your Location Phone Repair Other Repair Business

Google Authorized Repair on Pixel Smartphones

If it's not authorized, it's not fixed.

- ✓ We use OEM parts, tools, training, and equipment provided by Google.
- ✓ Most repairs done in 45 minutes or less.
- ✓ All repairs backed by a 1 Year Warranty.

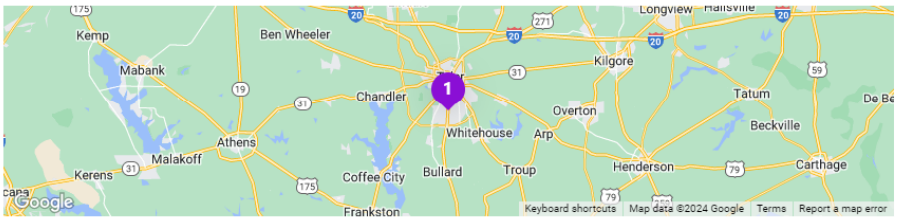
Find Your Repair Options

Source: https://www.ubreakifix.com/google-repair/google-pixel-repair

Find a nearby store

Enter ZIP code or city
Tyler, TX, USA

Use current location



1 store near you

1 Tyler
6721 S. Broadway Suite 100, Tyler
Next available **today, 5:00pm**

Source: https://www.asurion.com/repairs/tech/locations/tyler/

48. Google and uBreakiFix teamed up to offer free repairs to those impacted by Hurricane Florence.

49. uBreakiFix has fixed geographical locations. They are "regular" and "established" because they operate in a "steady, uniform, orderly, and methodical manner" and are sufficiently permanent. These stores are "of the defendant" because Google has contractual rights with

uBreakiFix—the only authorized walk-in repair centers in the United States. Google also ratifies these facilities through its advertising of them through its website.

50. Google also has branded, mail-in repair service that is contracted with a company called KMT Wireless, LLC, d/b/a Cynergy Hitech. Cynergy Hitech receives phones at its facility in Grapevine, Texas.

Find a repair partner

Use the table below to find available repair partners in your region. If there isn't an option available for your region, contact us.

Choose your country: United States

Region	Provider	Devices	Type of service	Repair type	Contact
United States	Google	All Pixel phones	Mail-in and walk-in	In-warranty and out-of-warranty	Google *
		Pixel Tablet	Mail-in	Out-of-warranty	Google
	Asurion or uBreakiFix	All Pixel phones	Mail-in and walk-in	Out-of-warranty	**Phone**: +1 877-320-2237 Asurion or uBreakiFix
		Pixel 6 and later, including Fold	Mail-in and walk-in	In-warranty and out-of-warranty	

*For the most up-to-date partner locations and options for your specific device or damage, refer to Google .

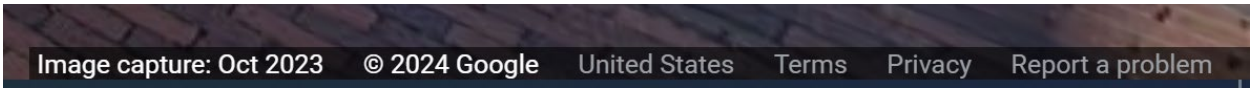
In US and Canada, replacement of parts or product service is made available for a minimum of 3 years after end of production for all Pixel devices through Google or its service providers.

Source: https://support.google.com/store/answer/7182296?hl=en

51. Google has operated and is currently operating its Google Maps Street View business and services in this Judicial District. For example, the image below shows the Google Maps Street View of the Eastern District of Texas courthouse in Marshall.

Source: https://www.google.com/maps/@32.5447184,-94.3668888,3a,75y,211.78h,88.91t/data=!3m6!1e1!3m4!1s1iGIaeAXPTpQQ3PT48kX1Q!2e0!7i16384!8i8192?entry=ttu

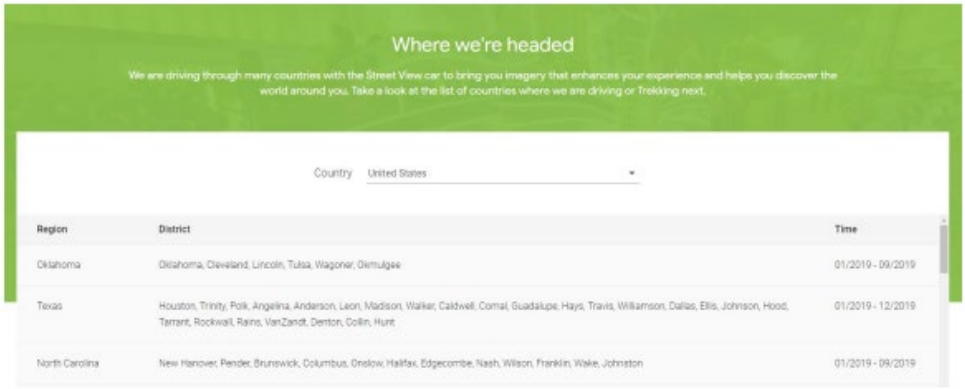
Furthermore, in the lower right-hand corner of the Google Street View above, the image is credited to Google and states that it was captured in December 2018.



52. Google also operates a Street View car in and around this Judicial District in order to provide the Google Maps Street View service.[14]

53. In addition to the above Google Street View image, Google operates and continues to operate a fleet of Google Street View vehicles in this Judicial District, including in the counties of Houston, Trinity, Polk, Angelina, Anderson, VanZandt, Denton, and Collin, as shown below.

[14] *See* https://www.google.com/streetview/explore/



Where we're headed

We are driving through many countries with the Street View car to bring you imagery that enhances your experience and helps you discover the world around you. Take a look at the list of countries where we are driving or Trekking next.

Country United States

Region	District	Time
Oklahoma	Oklahoma, Cleveland, Lincoln, Tulsa, Wagoner, Okmulgee	01/2019 - 09/2019
Texas	Houston, Trinity, Polk, Angelina, Anderson, Leon, Madison, Walker, Caldwell, Comal, Guadalupe, Hays, Travis, Williamson, Dallas, Ellis, Johnson, Hood, Tarrant, Rockwall, Rains, VanZandt, Denton, Collin, Hunt	01/2019 - 12/2019
North Carolina	New Hanover, Pender, Brunswick, Columbus, Onslow, Halifax, Edgecombe, Nash, Wilson, Franklin, Wake, Johnston	01/2019 - 09/2019

Source: https://www.google.com/streetview/explore/

54. Google provides its Google Express business and services to the residents of this Judicial District by advertising and inviting the residents of this Judicial District, then Defendant arranges for a delivery company to bring the goods and products purchased through the Google Express website to the residents of this Judicial District.[15] This service uses fixed geographical stores in this Judicial District. They are "regular" and "established" because they operate in a "steady, uniform, orderly, and methodical manner" and are sufficiently permanent. They are "of the defendant" because Google ratifies the stores (and selects products of the stores) through its website. Only information provided by Google through its service can be purchased, although the store may have other items for sale.

55. Google previously leased office space in this Judicial District for about 50 people through its Frisco, Texas office.

56. Google also provides services to businesses and schools in this Judicial District, including email services, word processing software, electronic file storage services, and video conferencing services. Google brands such services as "G Suite" services. Non-limiting examples of such businesses and schools include the Frisco Independent School District, as shown below.[16]

---

[15] *See* https://support.google.com/express/answer/4561693?hl=en
[16] http://schools.friscoisd.org/hs/lebanontrail/site/resources/google-apps-information

Google Classroom

Google Classroom is a web-based platform that integrates students' G Suite for Education accounts with Google Docs. Google Classroom saves time and paper and allows teachers to create classes, distribute assignments and communicate.

Google Classroom is already available on Frisco ISD devices. If students are accessing via a family-owned device, it may be helpful to download the app: iOS devices /Android devices.

Google Classroom Resources

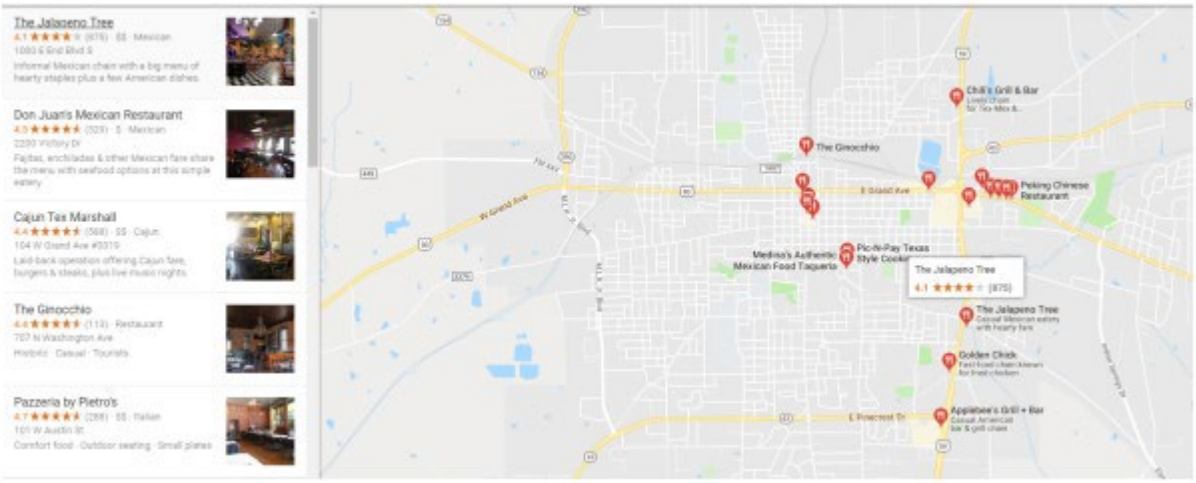
- **What is Google Classroom and How to Use It?**
- **Parents' Video Guide to Google Classroom**
- **Student Quick Google Classroom Reference**
- **Additional Google Classroom Resources**

Google Classroom Access & Log in:

- **At Home**
- **Via the Student Portal**

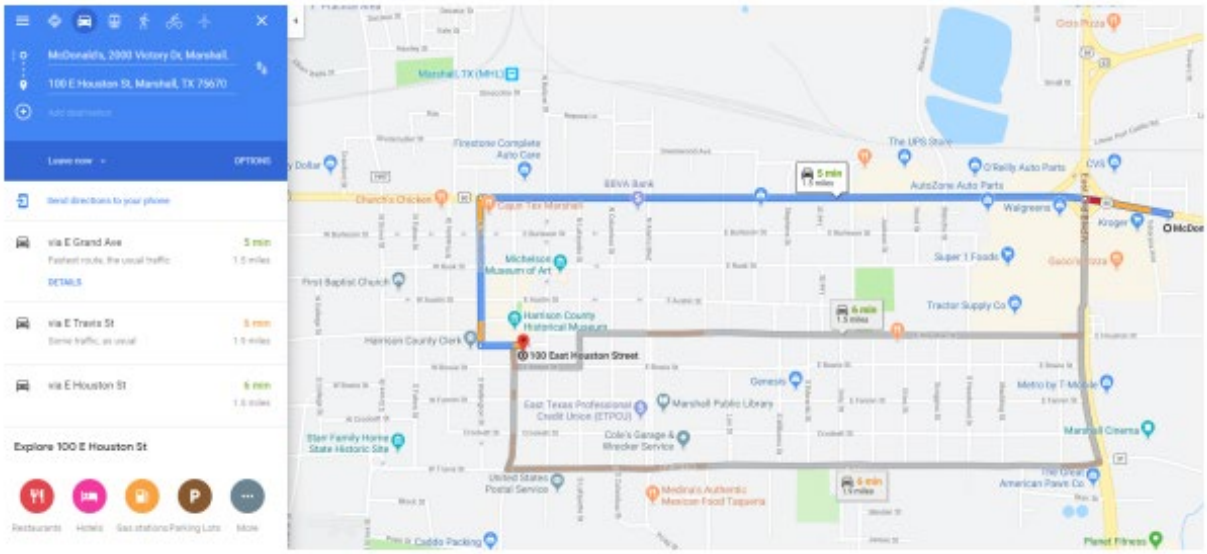
Source: https://www.friscoisd.org/departments/technology-and-media-services/technology-tools-support#q5

57. Google also provides advertising services to businesses in this Judicial District, including soliciting reviews of patrons that have visited a business in the Eastern District of Texas, as shown below.



Source: Product Testing at https://www.google.com/maps

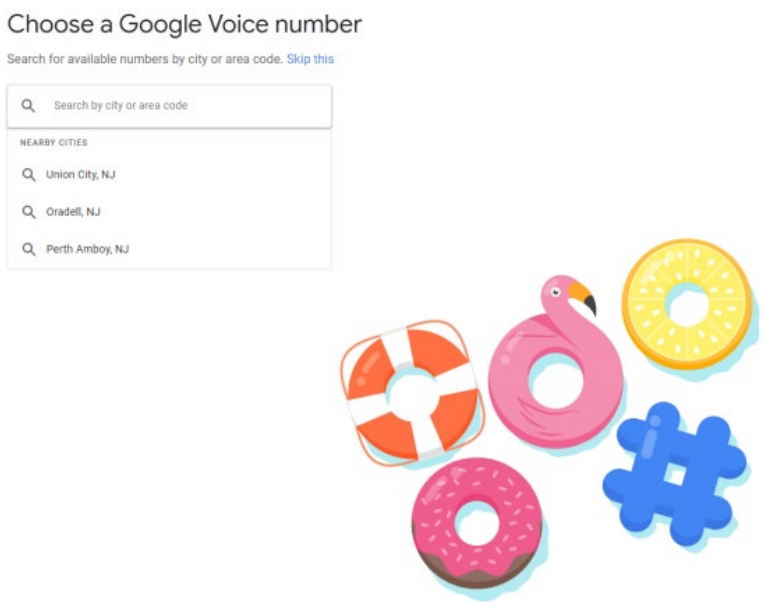
58. Google also monitors traffic conditions in this Judicial District. For example, traffic conditions between a McDonalds and the Federal Courthouse in Marshall, as shown below.



Source: Product Testing at https://www.google.com/maps

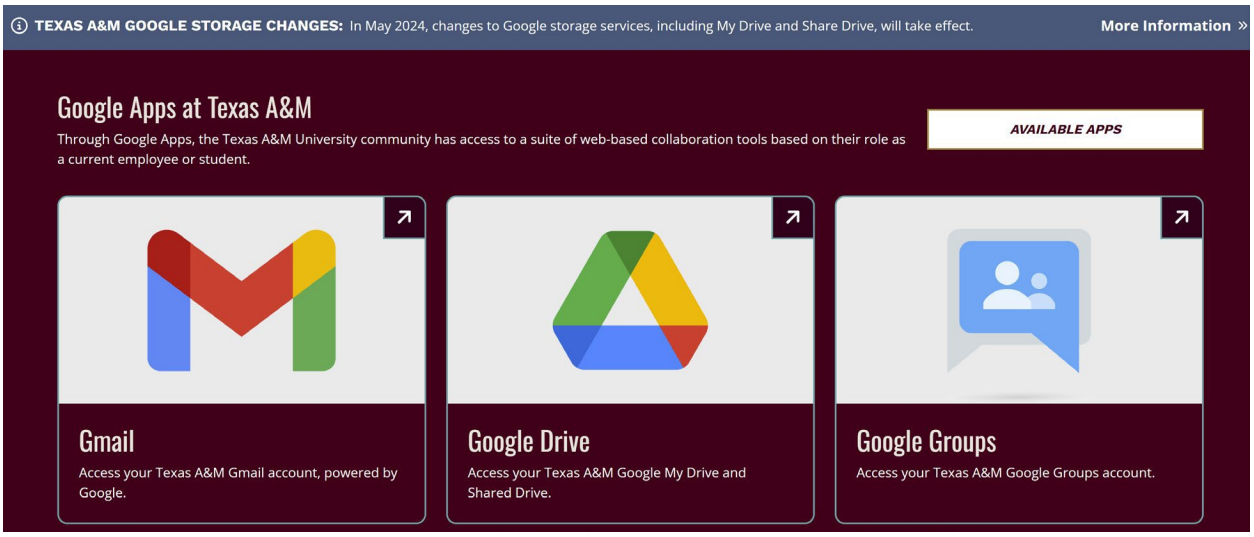
59. Separate and apart from its Google Fi mobile service, Google also provides telephone services to residents in this Judicial District through a product it calls Google Voice.[17]

[17] https://voice.google.com/u/0/signup



Source: https://voice.google.com/u/0/signup

60. Google provides Software-as-a-Service applications, including email and server space, to Texas public universities. Non-limiting examples of such universities are Texas A&M University (which has facilities in this Judicial District) and Texas A&M Commerce (located in this Judicial District), as shown below.



Source: http://google.tamu.edu/

```
Welcome Lions to your new LeoMail 2.0 found in your myLEO homepage located
at myLEO.tamu-commerce.edu.
We hope you take some time to look through your new student email. As a
reminder the new email is a gmail platform and share many features that a
regular gmail account has.

In addition to email, you will have the ability to build your own contacts
list and use the built in calendar for planning and organizing.
The most asked question has revolved around the ability to sync this email
account with your mobile or smart phone device. The answer is ³yes². The
Portal Implementation Team is working on getting both the email and your NEW
myLEO account connected in an application that will be available in June.
```

Source: http://mailman.tamuc.edu/pipermail/students/2012-May/004325.html

Other Google Presence in the State

61. Google also has a pervasive connection to the State of Texas through multiple commercial activities.

62. Google has purchased land in Midlothian, Texas where it is currently constructing a $600 million data center.[18]

63. Since 2007, Google has employed "hundreds" of employees in Texas, including in Austin, Texas.[19]

64. Google has at least one current office located in Austin, on North MoPac Expressway,[20] and additional office locations at University Park and Austin Children's Museum.[21]

---

[18] *See* https://www.dallasnews.com/business/real-estate/2019/06/14/google-s-massive-600m-data-center-takes-shape-in-ellis-county-as-tech-giant-ups-texas-presence/

[19] According to Gerardo Interiano, Google's public affairs and government relations manager, in a statement. *See* http://www.statesman.com/business/google-lease-200-000-square-feet-newdowntown-austin-tower/SANZSa3du8QQ4k8ytOC2rJ/

[20] *See* https://www.google.com/intl/en/about/locations/?region=north-america

[21] *See* http://www.statesman.com/business/google-lease-200-000-square-feet-new-downtownaustin-tower/SANZSa3du8QQ4k8ytOC2rJ/

65. Google has leased over 200,000 square feet of office space in Austin, Texas at 500 West 2nd Street.[22]

66. Google has, as of May 2024, job postings for Austin, Texas, Red Oak, Texas, Addison, Texas, Dallas, Texas, Houston, Texas, Midlothian, Texas, and Austonio, Texas (129 postings) including positions such as:

- Power Monitoring Execution Engineer, Google Data Center (Midlothian, TX)
- Network Implementation Engineer, Global Network Delivery (Addison, TX)
- Senior Finance Manager, Data Center Equipment (Austin, TX)
- Data Center Operations Facility Technician, Generators (Red Oak, TX)
- Field Solutions Developer II, Generative AI, Google Cloud (Houston, TX)
- Chrome Enterprise Premium Specialist (Austonio, TX)

67. Upon information and belief, Defendant has at least eleven (11) entities registered in Texas, including:

- GOOGLE LLC
- GOOGLE ACQUISITION HOLDING, INC.
- GOOGLE COMPARE AUTO INSURANCE SERVICES INC.
- GOOGLE COMPARE CREDIT CARDS INC.
- GOOGLE COMPARE MORTGAGES INC.
- GOOGLE FIBER INC.
- GOOGLE FIBER NORTH AMERICA INC.
- GOOGLE FIBER TEXAS, LLC

[22] *See* http://www.statesman.com/business/google-lease-200-000-square-feet-new-downtownaustin-tower/SANZSa3du8QQ4k8ytOC2rJ/

- GOOGLE INC.
- GOOGLE NORTH AMERICA INC.
- GOOGLE PAYMENT CORP.

68. Google has provided, currently provides, and is currently offering to provide its Google Fiber services to the residents of Austin, Texas and San Antonio, Texas.[23]

69. Google has invested $200,000,000 in the Spinning Spur Wind Farm Project in Oldham County, Texas.[24]

70. Google acquired Waze in 2013,[25] and Google's Waze traffic app partners with cities and businesses in Texas, non-limiting examples of which include the Waze partnership with the City of Fort Worth to provide constant traffic data to the city.[26] Another non-limiting example includes the Waze partnership with the Genesis Group in Tyler to decrease emergency response times.[27]

71. This Court has previously found that Google maintains a regular and established place of business in this Judicial District. *AGIS Software Dev. LLC v. Google LLC*, Case No. 2:19-CV-00361-JRG, Dkt. 378.

PATENTS-IN-SUIT

72. On January 3, 2012, the United States Patent and Trademark Office duly and legally issued U.S. Patent No. 8,090,025 (the "'025 Patent") entitled "Moving-Picture Coding Apparatus Method and Program, and Moving-Picture Decoding Apparatus, Method and Program." On

[23] *See* https://fiber.google.com/cities/austin/ and https://fiber.google.com/cities/sanantonio/
[24] *See* https://venturebeat.com/business/googles-tosses-200m-at-spinning-spur-wind-project-to-bring-its-green-power-to-2-gigawatts/
[25] *See* https://techcrunch.com/2013/06/11/its-official-google-buys-waze-giving-a-social-data-boost-to-its-location-and-mapping-business/
[26] *See* https://dallasinnovates.com/fort-worth-waze-partner-ease-traffic-woes/
[27] *See* https://genesisworld.com/the-genesis-group-joins-waze-connected-citizens-program/

October 4, 2022, the United States Patent and Trademark Office duly and legally issued a Certificate of Correction to the ’025 Patent. A true and correct copy of the ’025 Patent is attached hereto as Exhibit A.

73. On May 29, 2018, the United States Patent and Trademark Office duly and legally issued U.S. Patent No. 9,986,303 (the “’303 Patent”) entitled “Video Image Coding Data Transmitter, Video Image Coding Data Transmission Method, Video Image Coding Data Receiver, and Video Image Coding Data Transmission and Reception System.” A true and correct copy of the ’303 Patent is attached hereto as Exhibit B.

74. On February 26, 2019, the United States Patent and Trademark Office duly and legally issued U. S. Patent No. 10,218,995 (the “’995 Patent”) entitled “Moving Picture Encoding System, Moving Picture Encoding Method, Moving Picture Encoding Program, Moving Picture Decoding System, Moving Picture Decoding Method, Moving Picture Decoding Program, Moving Picture Reencoding System, Moving Picture Reencoding Method, Moving Picture Reencoding Program.” A true and correct copy of the ’995 Patent is attached hereto as Exhibit C.

75. On May 26, 2015, the United States Patent and Trademark Office duly and legally issued U.S. Patent No. 9,042,448 (the “’448 Patent”) entitled “Moving Picture Encoding System, Moving Picture Encoding Method, Moving Picture Encoding Program, Moving Picture Decoding System, Moving Picture Decoding Method, Moving Picture Decoding Program, Moving Picture Reencoding System, Moving Picture Reencoding Method, and Moving Picture Reencoding Program.” A true and correct copy of the ’448 Patent is attached hereto as Exhibit D.

76. On July 24, 2012, the United States Patent and Trademark Office duly and legally issued U.S. Patent No. 8,230,101 (the “’101 Patent”) entitled “Server Device for Media, Method for Controlling Server for Media, and Program.” A true and correct copy of the ’101 Patent is

attached hereto as Exhibit E.

77. On September 28, 2010, the United States Patent and Trademark Office duly and legally issued U. S. Patent No. 7,804,891 (the "'891 Patent") entitled "Device and Method for Judging Communication Quality and Program Used for the Judgment." A true and correct copy of the '891 Patent is attached hereto as Exhibit F.

78. ACT is the sole and exclusive owner of all right, title, and interest in the '025 Patent, the '303 Patent, the '995 Patent, the '448 Patent, the '101 Patent, and the '891 Patent (collectively, the "Patents-in-Suit") and holds the exclusive right to take all actions necessary to enforce its rights to the Patents-in-Suit, including the filing of this patent infringement lawsuit. ACT also has the right to recover all damages for past, present, and future infringement of the Patents-in-Suit.

FACTUAL ALLEGATIONS

79. The '025 Patent generally relates to efficient methods of video encoding and decoding using motion compensation. The technology described in the '025 Patent was developed by Satoru Sakazume of Victor Company of Japan, Ltd.

80. The '303 Patent generally relates to technology that allows for the efficient transmission and reception of two different resolutions of video data. The technology described in the '303 Patent was developed by Hideki Takehara and Motoharu Ueda of JVC Kenwood Corporation.

81. The '995 Patent generally relates to hierarchical encoding that implements a process for super-resolution enlargement of video signals. The technology described in the '995 Patent was developed by Satoru Sakazume of JVC Kenwood Corporation.

82. The '448 Patent generally relates to hierarchical encoding that implements a process for super-resolution enlargement of video signals. The technology described in the '448

Patent was developed by Satoru Sakazume of JVC Kenwood Corporation.

83. The ’101 Patent generally relates to a server for media capable of smoothly dealing with a large amount of digital contents. The technology described in the ’101 Patent was developed by Satoru Sekiguchi, Yoshio Sonoda, Isao Nakamura, Masamichi Furukawa, Yoshihia Mashita, Tomoaki Yoshida, and Masahito Watanabe of Kabushiki Kaisha Kenwood.

84. The ’891 Patent generally relates to judging communication quality in a communication system, and a program for causing a computer to execute the judgment. The technology described in the ’891 Patent was developed by Taichi Majima of Kabushiki Kaisha Kenwood.

85. In September 2015, Defendant founded the Alliance for Open Media (“AOM”), “an open-source project that will develop next-generation media formats, codecs and technologies” and is listed as a “founding member.”[28] Google’s Head of Strategy and Partnerships for Chrome Media, Matt Frost provided a supporting statement for the goal of AOM:

> "Google launched the WebM Project in 2010 in the belief that web video innovation was too slow and too closed, and that broad collaboration — in the open — would fix both problems. The Alliance for Open Media is a big leap forward for these core philosophies, and we're gratified that our AOMedia partners share this vision. Our combined strength, resources and expertise will drive the next generation of web media experiences much further and faster than WebM can do alone," said Matt Frost, Head of Strategy and Partnerships, Chrome Media.

Source: https://aomedia.org/press%20releases/alliance-to-deliver-next-generation-open-media-formats/

[28] https://aomedia.org/press%20releases/alliance-to-deliver-next-generation-open-media-formats/

Matt Frost is currently listed as the Chair of AOM's Steering Committee:

> The AOMedia Steering Committee is responsible for the leadership and general management of AOMedia, including setting its mission, vision and objectives. The Steering Committee consists of the following representatives:

Member	Organization
John Elovson	Amazon
Krasimir Kolarov	Apple
Dale Mohlenhoff (*Finance*)	Cisco
Matt Frost (Chair)	Google
Iole Moccagatta	Intel
David Ronca	Meta
Steven Lees	Microsoft
Daniel Nazer	Mozilla
Anne Aaron (*Vice-Chair, Communications / Membership*)	Netflix
Frans Sijstermans	nVIDIA
Kwang Pyo Choi	Samsung
Shan Liu	Tencent

Source: https://aomedia.org/about/

86. Defendant has infringed and continues to infringe one or more of the Patents-in-Suit by making, using, selling, offering to sell, and/or importing, and by actively inducing others to make, use, sell, offer to sell, and/or import products, including mobile devices and chipsets thereof, that implement the technology claimed by the Patents-in-Suit. For example, the Accused Products include, but are not limited to, (i) Defendant's Google Pixel smartphone and tablet products, Google Chromebook laptop products, Google Pixel Slate, Google Chrome, Google Chromecast, Android/Google TV, Youtube, Google Meet, and Google Duo that encode and/or decode digital video using either an AV1 codec with Tensor processors and/or software-based AV1 codec (including through Android OS updates) with respect to the '025, '303, '995, and '448

Patents; (ii) Youtube, Defendant's Google Cloud Content Delivery Network (CDN), and Defendant's Google Smart Home with respect to the '101 Patent; and (iii) Defendant's Google Pixel smartphone products, which implement 5G NR technology with respect to the '891 Patent.

87. Google has had actual notice of the '025 and '303 Patents from related prior litigations accusing products with similar AV1 functionalities involving direct competitors of Defendant.

88. Defendant has been on actual notice of the '995 Patent and the '448 Patent and Defendant's infringement thereof at least as of June 3, 2013, when it was cited during prosecution of U.S. Patent No. 8,635,357, entitled "Dynamic selection of parameter sets for transcoding media data," assigned to Google LLC. Specifically, during prosecution, the Examiner cited to U.S. Patent Publication No. 2011/0075734 A1, which is the published patent application number of the '448 Patent, of which the '995 patent application is a divisional patent application.

89. Defendant has had actual notice of the '101 and '891 Patents, at least as of the filing date of this First Amended Complaint.

90. ACT has, at all times, complied with the marking provisions of 35 U.S.C. § 287 with respect to the Asserted Patents.

COUNT I
(Infringement of the '025 Patent)

91. Paragraphs 1 through 90 are incorporated by reference as if fully set forth herein.

92. ACT has not licensed or otherwise authorized Defendant to make, use, offer for sale, sell, or import any products that embody the inventions of the '025 Patent.

93. Defendant has and continues to directly infringe the '025 Patent, either literally or under the doctrine of equivalents, without authority and in violation of 35 U.S.C. § 271, by making, using, offering to sell, selling, and/or importing into the United States products that satisfy each

and every limitation of one or more claims of the ’025 Patent. Such products include at least Google Pixel Smartphone products compliant with the AV1 and/or SVT-AV1 Standards including, but not limited to, the Pixel 6, Pixel 6 Pro, Pixel 6a, Pixel 7, Pixel 7 Pro, Pixel 7a, Pixel Fold, Pixel 8, Pixel 8 Pro, and Pixel 8a, Google Chromebook laptop products, Google Pixel Slate, Google Chrome, Google Chromecast, Android/Google TV, Youtube, Google Meet, and Google Duo (the ’025 Accused Products), which practice a moving-picture decoding method comprising the steps of: demultiplexing coded data from an input signal based on a specific syntax structure, the input signal being obtained by multiplexing a coded bitstream obtained by predictive coding, border motion-vector data and post-quantization data obtained by quantization in the predictive coding, the coded bitstream obtained by producing and encoding a residual picture that is a residual signal between a picture to be coded that is an input moving-picture video signal to be subjected to coding and a predictive picture produced from a reference picture that is a local decoded video signal for each of a plurality of rectangular zones, each composed of a specific number of pixels, into which a video area of the moving-picture video signal is divided, obtaining a boundary condition of each of a plurality of borders between the rectangular zones and another plurality of rectangular zones adjacent to the rectangular zones, finding a border, of the reference picture, having a boundary condition that matches the boundary condition, by motion-vector search in the reference picture, and generating the border motion-vector data that is data on a motion vector from a border of the rectangular zone in the picture to be coded to the border of the reference picture thus found, defining a boundary condition of a border that corresponds to the border motion-vector data, from the reference picture based on the border motion-vector data, and generating an estimated video signal in each rectangular zone in the picture to be coded, that satisfies Poisson’s Equation, thus producing the predictive picture; performing entropy decoding to the data thus demultiplexed to

generate, at least, the post-quantization data, the border motion-vector data and parameter data required for constructing a specific syntax structure; performing inverse-quantization to the post-quantization data to generate post-quantization orthogonal transform coefficients data; performing inverse-orthogonal transform to the post-quantization orthogonal transform coefficients data to produce a decoded residual picture of one video area; defining a boundary condition of a border that corresponds to the border motion-vector data, from the reference picture based on the border motion-vector data, and generate an estimated video signal in each rectangular zone in the picture to be coded, that satisfies Poisson's Equation, thus producing a first predictive picture; combining the first predictive picture and the decoded residual picture to generate a decoded moving-picture signal; and storing the decoded moving-picture signal for at least one picture as a reference picture.

94. For example, Defendant has and continues to directly infringe at least claim 10 of the '025 Patent by making, using, offering to sell, selling, and/or importing into the United States the '025 Accused Products.

95. The '025 Accused Products demultiplex coded data from an input signal based on a specific syntax structure, the input signal being obtained by predictive coding, border motion-vector data and post-quantization data obtained by quantization in the predictive coding:

Which Pixel is right for you?

Pixel 8 Pro	Pixel 8	Pixel 8a
The best of Google. Even more pro.	Powerful in every way. Helpful every day.	Colorful. Powerful. AI-full.
From $999	From $699	From $499
or $27.75/month with 36-month financing*	or $19.42/month with 36-month financing*	or $20.79/month with 24-month financing*
Buy Learn more >	Buy Learn more >	Pre-order Learn more >

Display

6.7"	6.2"	6.1"
Super Actua display[5]	Actua display[5]	Actua display[6]
1 - 120 Hz	60 - 120 Hz	60 - 120 Hz
Up to 1600 nits (HDR) and up to 2400 nits (peak brightness)[14]	Up to 1400 nits (HDR) and up to 2000 nits (peak brightness)[14]	Up to 1400 nits (HDR) and up to 2000 nits (peak brightness)[14]

Rear camera

Triple rear camera system:	Dual rear camera system:	Dual rear camera system:
Wide lens, ultrawide lens,[7] telephoto lens	Wide lens, ultrawide lens[7]	Wide lens, ultrawide lens[7]

Processor

G3	G3	G3
Google Tensor G3	Google Tensor G3	Google Tensor G3

Google Tensor G3 processor specifications

	Google Tensor G3	Google Tensor G2	Google Tensor
CPU	1x Arm Cortex-X3 (2.91GHz) 4x Arm Cortex-A715 (2.37GHz) 4x Arm Cortex-A510 (1.70GHz)	2x Arm Cortex-X1 (2.85GHz) 2x Arm Cortex-A78 (2.35GHz) 4x Arm Cortex-A55 (1.80GHz)	2x Arm Cortex-X1 (2.80GHz) 2x Arm Cortex-A76 (2.25GHz) 4x Arm Cortex-A55 (1.80GHz)
GPU	Arm Mali-G715 (MP7 estimated)	Arm Mali-G710 MP7	Arm Mali-G78 MP20
Caches	Unknown	4MB CPU L3 8MB system level	4MB CPU L3 8MB system level
Machine Learning	Third-gen Tensor Processing Unit	Next-gen Tensor Processing Unit	Tensor Processing Unit
Media Decode	H.264, H.265, VP9, AV1	H.264, H.265, VP9, AV1	H.264, H.265, VP9, AV1
Modem	4G LTE 5G sub-6Ghz and mmWave	4G LTE 5G sub-6Ghz and mmWave	4G LTE 5G sub-6Ghz and mmWave
Process	Samsung 4nm (expected)	Samsung 5nm	Samsung 5nm

Source: https://store.google.com/us/category/phones?hl=en-US
https://www.androidauthority.com/google-tensor-g3-explained-3324692/

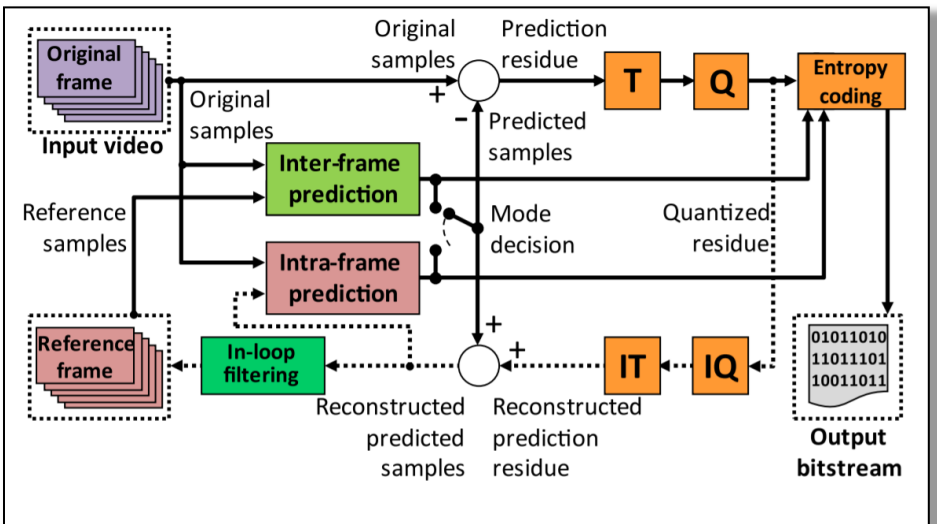


FIGURE 1. Block diagram of a typical hybrid video encoder.

Source: https://ieeexplore.ieee.org/ielx7/8784029/9314963/09536216.pdf

96. The coded bitstream in the ’025 Accused Products is obtained by producing and encoding a residual picture that is a residual signal between a picture to be coded that is an input moving-picture video signal to be subjected to coding and a predictive picture produced from a reference picture that is a local decoded video signal for each of a plurality of rectangular zones, each composed of a specific number of pixels, into which a video area of the moving-picture video signal is divided:

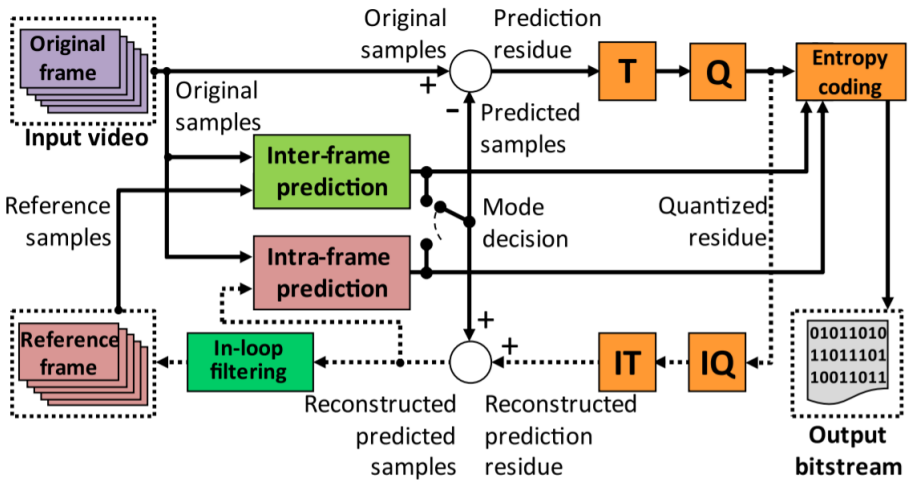


FIGURE 1. Block diagram of a typical hybrid video encoder.

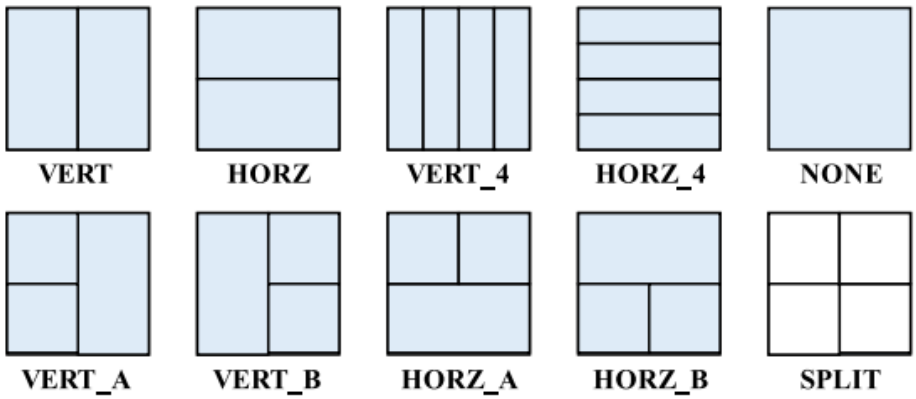


FIGURE 2. AV1 10-way block partition tree structure.

Source: https://ieeexplore.ieee.org/ielx7/8784029/9314963/09536216.pdf

97. The ’025 Accused Products obtain a boundary condition of each of a plurality of borders between the rectangular zones and another plurality of rectangular zones adjacent to the rectangular zones, find a border, of the reference picture, having a boundary condition that matches the boundary condition, by motion-vector search in the reference picture, and generate the border motion-vector data that is data on a motion vector from a border of the rectangular zone in the picture to be coded to the border of the reference picture thus found, by using the motion estimation process for a block and locating the pixel values at the border between the current block and the neighboring block. Border motion-vector data is generated when a boundary condition in the reference frame matches the boundary condition in the current frame, and the block motion estimation algorithm uses a comparison of these boundary conditions to generate motion vectors:

> ***C. INTER PREDICTION***
> In inter-frame prediction, the block is predicted from samples belonging to previously encoded frames. Both AV1 and VVC use Motion Estimation (ME) and Motion Compensation (MC) algorithms in addition to motion vector prediction tools to reduce the amount of lateral data. Both video formats allow block sizes from 128×128 to 4×4 in inter prediction. VVC and AV1 can evaluate 28 and 22 block sizes, respectively, in function of the difference between their frame partition processes.

Source: https://ieeexplore.ieee.org/document/8296419

98. The ’025 Accused Products define a boundary condition of a border that corresponds to the border motion-vector data, from the reference picture based on the border motion-vector data, and generate an estimated video signal in each rectangular zone in the picture to be coded, that satisfies Poisson’s Equation, thus producing the predictive picture. For example, the estimated signal generation process in AV1 and/or SVT-AV1 satisfies Poisson’s Equation via the use of smoothing algorithms in Overlapped Block Motion Compensation (“OMBC”). The

process involves finding predicted pixels of a block in steady state (that minimizes the residual). The estimated video signal is used to produce a predictive picture (*e.g.*, predictive sample):

(see Fig.2(b) for a 32-tap example) and formulated as

$$w_K(k) = \frac{1}{2}sin(\frac{\pi}{2K}(k+\frac{1}{2})) + \frac{1}{2}, k = 0, 1, ..., K-1 \quad (1)$$

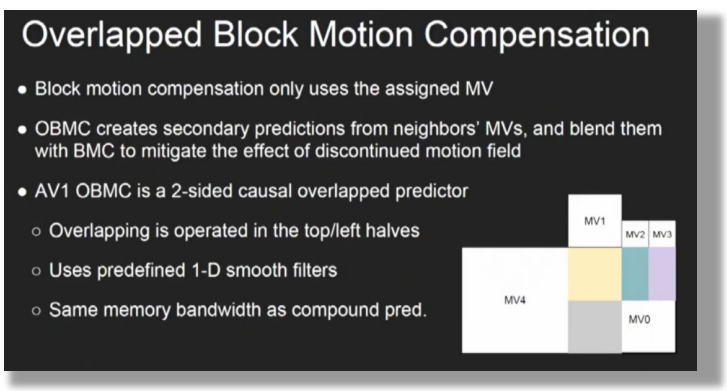
is applied to every column (see Fig.2(a)) of the overlapping region, and updates $p_{obmc}(x, y)$ as

$$w_{\frac{N}{2}}(y)p_{obmc}(x, y) + (1 - w_{\frac{N}{2}}(y))p_i(x, y). \quad (2)$$

This filter approximately averages the predictions at the common edge, and gradually reduces the influence of the new prediction p_i until it vanishes at the mid-line of the current block, because the conventional block matching p_0 often works best for pixels in the center. Then we move on to the second stage to exploit predictors of the left neighbors. Likewise, 1-D filtering will be performed on top of the $p_{obmc}(x, y)$ updated after the first phase: (1) the overlapping region for each left neighbors will be on the right side of the common edge, e.g. the shaded area for p_4 in Fig.1(b); (2) we apply the 1-D filter in the horizontal direction, i.e.

$$p_{obmc}(x, y) := w_{\frac{M}{2}}(x)p_{obmc}(x, y) + (1 - w_{\frac{M}{2}}(x))p_i(x, y). \quad (3)$$

Source: https://ieeexplore.ieee.org/document/8296419



Source: https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

99. The '025 Accused Products perform entropy decoding to the data thus demultiplexed to generate, at least, the post-quantization data, the border motion-vector data, and parameter data required for constructing a specific syntax structure:

E. ENTROPY CODING

The entropy coding processes the symbols (quantized coefficients and lateral data) to reduce their statistical redundancy by applying lossless algorithms.

AV1 uses a symbol-to-symbol adaptive multi-symbol arithmetic coder with the probability being updated every new symbol. Each syntax element in AV1 is a member of an alphabet of N elements, and a context consists of a set of N probabilities together with a small count to facilitate fast early adaptation [2].

D. TRANSFORMS AND QUANTIZATION

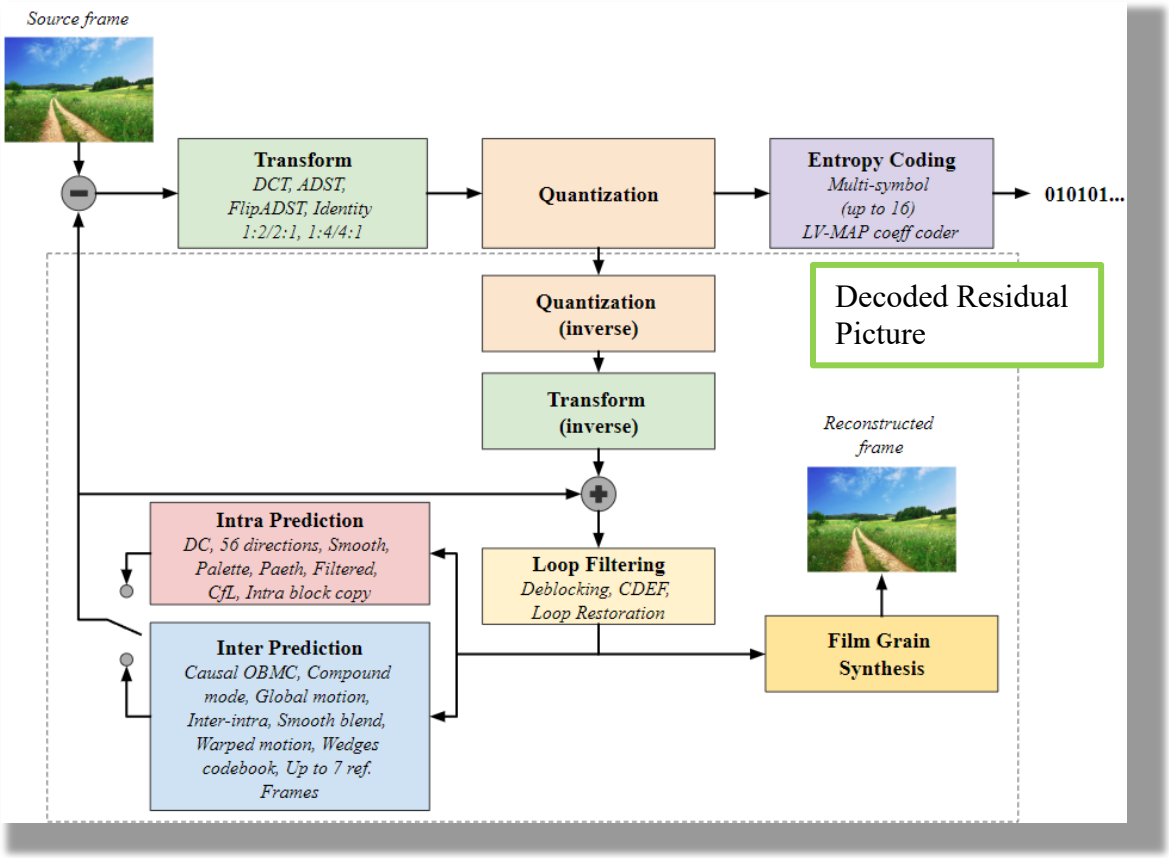
The prediction error, or the residues, between the intra and inter prediction and the original blocks are processed by the transform module (T module, in Fig. 1), which converts the values from the spatial domain to the frequency domain. Then, the quantization step (Q module in Fig. 1) is applied to the transformed coefficients to attenuate or eliminate values associated with spectral components that are not perceptually relevant for the human visual system.

Source: https://ieeexplore.ieee.org/ielx7/8784029/9314963/09536216.pdf

100. The ’025 Accused Products perform inverse-quantization to the post-quantization data to generate post-quantization orthogonal transform coefficients data, and perform inverse-orthogonal transform to the post-quantization orthogonal transform coefficients data to produce a decoded residual picture of one video area.

101. The ’025 Accused Products define a boundary condition of a border that corresponds to the motion-vector data, from the reference picture based on the border motion-vector data, and generate an estimated video signal in each rectangular zone in the picture to be coded, that satisfied Poisson’s Equation, thus producing a first predictive picture.

102. The ’025 Accused Products combine the first predictive picture and the decoded residual picture to generate a decoded moving-picture signal:



Source: https://wenxiaoming.github.io/2019/03/02/The-overview-of-AV1-coding/

7.14. Loop filter process

7.14.1. General

Input to this process is the array CurrFrame of reconstructed samples.

Output from this process is a modified array CurrFrame containing deblocked samples.

Source: https://aomediacodec.github.io/av1-spec/av1-spec.pdf, Page 307

103. The ’025 Accused Products store the decoded moving-picture signal for at least one picture as a reference picture, by updating the set of reference frames.

104. Defendant has and continues to directly infringe at least claim 10 of the ’025 Patent by making, using, offering to sell, selling, and/or importing into the United States products that

implement AV1 and/or SVT-AV1 standards, such as the ’025 Accused Products.

105. Google has indirectly infringed and continues to indirectly infringe one or more claims of the ’025 Patent, as provided by 35 U.S.C. § 271(b), by inducing infringement by others, such as Google’s customers and end-users, in this Judicial District and elsewhere in the United States. For example, Google’s customers and end-users directly infringe, either literally or under the doctrine of equivalents, through their use of the inventions claimed in the ’025 Patent. Google induces this direct infringement through its affirmative acts of manufacturing, selling, distributing, and/or otherwise making available the ’025 Accused Products, and providing instructions, documentation, and other information to customers and end-users suggesting that they use the ’025 Accused Products in an infringing manner, including technical support, marketing, product manuals, advertisements, and online documentation.[29] Because of Google’s inducement, Google’s customers and end-users use the ’025 Accused Products in a way Google intends and they directly infringe the ’025 Patent. Google performs these affirmative acts with knowledge of the ’025 Patent and with the intent, or willful blindness, that the induced acts directly infringe the ’025 Patent.

106. Google has indirectly infringed and continues to indirectly infringe one or more claims of the ’025 Patent, as provided by 35 U.S.C. § 271(c), by contributing to direct infringement by others, such as customers and end-users, in this Judicial District and elsewhere in the United States. Google’s affirmative acts of selling and offering to sell the ’025 Accused Products in this Judicial District and elsewhere in the United States and causing the ’025 Accused Products to be manufactured, used, sold, and offered for sale contribute to others’ use and manufacture of the

[29] *See, e.g.*, https://guidebooks.google.com/home;
https://pixel.withgoogle.com/?hl=en&country=US;
https://pixel.withgoogle.com/support/Pixel?hl=en&country=US&dark=0;
https://www.youtube.com/@Googlehelp/playlists

Accused Products, such that the ’025 Patent is directly infringed by others.[30] The accused components within the Accused Products including, but not limited to, software manufactured by Google, are material to the invention of the ’025 Patent, are not staple articles or commodities of commerce, have no substantial non-infringing uses, and are known by Google to be especially made or adapted for use in the infringement of the ’025 Patent. Google performs these affirmative acts with knowledge of the ’025 Patent and with intent, or willful blindness, that they cause the direct infringement of the ’025 Patent.

107. Google’s infringement of the ’025 Patent is and has been willful. Google was on notice of the existence of the ’025 Patent and its infringement thereof, or has been willfully blind as to the existence of the ’025 Patent and its infringement thereof. As one example, Google is a founding member of the Alliance for Open Media, the organization that publishes the AV1 Specification, as of the time of its inception in 2015.[31] Google’s Head of Strategy and Partnerships for Chrome Media, Matt Frost, is currently listed as the Chair of AOM’s Steering Committee. The Alliance for Open Media’s stated goal was to create a video codec that was free of patent licensing obligations associated with prior video codecs.[32] For example, Google’s YouTube product previously used a video codec called HEVC, and Google was motivated to avoid HEVC licensing fees by developing AV1 through the Alliance for Open Media.[33] The Alliance for Open Media, including Google, conducted a “comprehensive evaluation of the video codec patent landscape and performance of patent due diligence by world-class codec engineers and legal professionals

[30] *See, e.g.*, https://store.google.com/magazine/support?hl=en-US
[31] https://aomedia.org/about/
[32] https://aomedia.org/press%20releases/alliance-to-deliver-next-generation-open-media-formats/
[33] https://arstechnica.com/information-technology/2015/09/microsoft-google-amazon-others-aim-for-royalty-free-video-codecs/

during the development stage."[34] Upon information and belief, this "patent due diligence" either uncovered the existence of the '025 Patent and Google's infringement thereof, or should have uncovered the existence of the '025 Patent and Google's infringement thereof. Google could not have reasonably believed that the development of the AV1 video codec could not infringe any valid patent claims, including those of the '025 Patent.

108. Upon information and belief, Defendant had actual knowledge of the '025 Patent from related prior litigations accusing products with similar AV1 functionalities involving direct competitors of Defendant.

109. ACT has suffered damages as a result of Defendant's direct, indirect, and willful infringement of the '025 Patent in an amount to be proved at trial.

COUNT II
(Infringement of the '303 Patent)

110. Paragraphs 1 through 90 are incorporated by reference as if fully set forth herein.

111. ACT has not licensed or otherwise authorized Defendant to make, use, offer for sale, sell, or import any products that embody the inventions of the '303 Patent.

112. Defendant has and continues to directly infringe the '303 Patent, either literally or under the doctrine of equivalents, without authority and in violation of 35 U.S.C. § 271, by making, using, offering to sell, selling, and/or importing into the United States products that satisfy each and every limitation of one or more claims of the '303 Patent. Such products include at least Google Pixel Smartphone Products compliant with the AV1 and/or SVT-AV1 Standards including, but not limited to, the Pixel 6, Pixel 6 Pro, Pixel 6a, Pixel 7, Pixel 7 Pro, Pixel 7a, Pixel Fold, Pixel 8, Pixel 8 Pro, Pixel 8a, Google Chromebook laptop products, Google Pixel Slate,

[34] https://aomedia.org/press%20releases/the-alliance-for-open-media-statement/

Google Chrome, Google Chromecast, Android/Google TV, Youtube, Google Meet, and Google Duo (the ’303 Accused Products) which include a video image coding data receiver comprising a processor and a memory unit having instructions stored which, when executed by the processor, cause the processor to perform operations comprising receiving basic video image coding data; decoding the received basic video image coding data so as to reproduce a video image; receiving supplementary video image coding data including a supplementary hierarchical picture whose coding order and display order are earlier by a factor of a group of pictures including an intra coded picture and a plurality of inter prediction coded pictures than those of a basic hierarchical picture included in the basic video image coding data, a basic hierarchy and a supplementary hierarchy being set in units of the group of pictures; acquiring basic video image coding data received before supplementary video image coding data that has been received at the moment; and reconstructing video image coding data from the basic video image coding data and the supplementary video image coding data.

113. For example, Defendant has and continues to directly infringe at least claim 1 of the ’303 Patent by making, using, offering to sell, selling, and/or importing into the United States the ’303 Accused Products.

114. The ’303 Accused Products are video image coding data receivers that include a processor and a memory.

115. The ’303 Accused Products are configured to receive and decode basic video image coding data, such as a bitstream of video at 720p resolution, and to decode that data to reproduce a video image.

116. The ’303 Accused Products are configured to receive supplementary video image coding data including a supplementary hierarchical picture, such as a bitstream of video at a 1080p

resolution.

117. The supplementary hierarchical picture's coding order and display order are earlier than those of a basic hierarchical picture by a factor of a group of pictures. For example, AV1 uses an S frame to switch to lower or higher frame rates:

Switch Frame
An inter frame that can be used as a point to switch between sequences. Switch frames overwrite all the reference frames without forcing the use of intra coding. The intention is to allow a streaming use case where videos can be encoded in small chunks (say of 1 second duration), each starting with a switch frame. If the available bandwidth drops, the server can start sending chunks from a lower bitrate encoding instead. When this happens the inter prediction uses the existing higher quality reference frames to decode the switch frame. This approach allows a bitrate switch without the cost of a full key frame.

Source: https://aomediacodec.github.io/av1-spec/av1-spec.pdf, at page 5

118. Each Group of Pictures includes an intra coded picture and a plurality of inter prediction coded pictures:

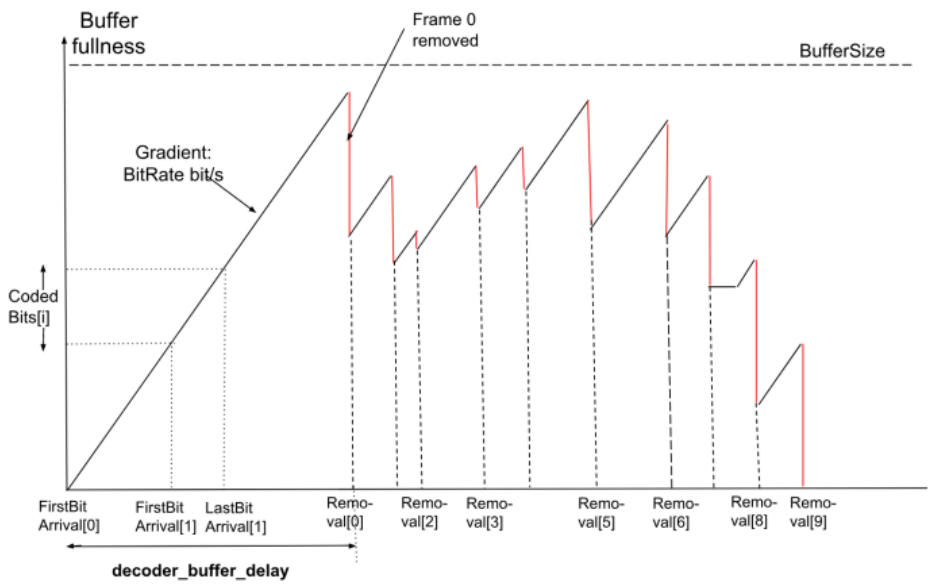
frame_type specifies the type of the frame:

frame_type	Name of frame_type
0	KEY_FRAME
1	INTER_FRAME
2	INTRA_ONLY_FRAME
3	SWITCH_FRAME

Source: https://aomediacodec.github.io/av1-spec/av1-spec.pdf, at page 150

119. The supplementary hierarchical picture's coding order and display order are earlier than the basic hierarchical picture because the received data is stored in a buffer before decoding:

PresentationTimes[i] is an array corresponding to the BufferPool [i] that holds the last presentation time for the decoded frame that is kept in the BufferPool [i].



Example of how the coded frame buffer fullness varies as data arrives from the stream, and is subsequently removed for decoding. Relevant timing points and values are indicated.

Source: https://aomediacodec.github.io/av1-spec/av1-spec.pdf, at page 654-55

Therefore, when an S frame switches from basic to supplementary video data, basic hierarchical pictures are still decoded and displayed out of the buffer.

120. The ’303 Accused Products are configured to acquire basic video image coding data from the buffer, which has been received before supplementary video image coding data that has been received at the moment of the switch in resolutions.

121. The ’303 Accused Products reconstruct video image coding data from the basic video image coding data and the supplementary video image coding data:

7.12.3. Reconstruct process

The reconstruct process is invoked to perform dequantization, inverse transform and reconstruction. This process is triggered at a point defined by a function call to reconstruct in the transform block syntax table described in section 5.11.35.

The inputs to this process are:

- a variable plane specifying which plane is being reconstructed,
- variables x and y specifying the location of the top left sample in the CurrFrame[plane] array of the current transform block,
- a variable txSz, specifying the size of the transform block.

The outputs of this process are reconstructed samples in the current frame CurrFrame.

Source: https://aomediacodec.github.io/av1-spec/av1-spec.pdf, at page 294

122. Google has indirectly infringed and continues to indirectly infringe one or more claims of the '303 Patent, as provided by 35 U.S.C. § 271(b), by inducing infringement by others, such as Google's customers and end-users, in this Judicial District and elsewhere in the United States. For example, Google's customers and end-users directly infringe, either literally or under the doctrine of equivalents, through their use of the inventions claimed in the '303 Patent. Google induces this direct infringement through its affirmative acts of manufacturing, selling, distributing, and/or otherwise making available the '303 Accused Products, and providing instructions, documentation, and other information to customers and end-users suggesting that they use the '303 Accused Products in an infringing manner, including technical support, marketing, product manuals, advertisements, and online documentation.[35] Because of Google's inducement, Google's customers and end-users use the '303 Accused Products in a way Google intends and they directly infringe the '303 Patent. Google performs these affirmative acts with knowledge of the '303 Patent

[35] *See, e.g.*, https://guidebooks.google.com/home; https://pixel.withgoogle.com/?hl=en&country=US; https://pixel.withgoogle.com/support/Pixel?hl=en&country=US&dark=0; https://www.youtube.com/@Googlehelp/playlists

and with the intent, or willful blindness, that the induced acts directly infringe the ’303 Patent.

123. Google has indirectly infringed and continues to indirectly infringe one or more claims of the ’303 Patent, as provided by 35 U.S.C. § 271(c), by contributing to direct infringement by others, such as customers and end-users, in this Judicial District and elsewhere in the United States. Google’s affirmative acts of selling and offering to sell the ’303 Accused Products in this Judicial District and elsewhere in the United States and causing the ’303 Accused Products to be manufactured, used, sold, and offered for sale contribute to others’ use and manufacture of the ’303 Accused Products, such that the ’303 Patent is directly infringed by others.[36] The accused components within the ’303 Accused Products including, but not limited to, software manufactured by Google, are material to the invention of the ’303 Patent, are not staple articles or commodities of commerce, have no substantial non-infringing uses, and are known by Google to be especially made or adapted for use in the infringement of the ’303 Patent. Google performs these affirmative acts with knowledge of the ’303 Patent and with intent, or willful blindness, that they cause the direct infringement of the ’303 Patent.

124. Google’s infringement of the ’303 Patent is and has been willful. Google was on notice of the existence of the ’303 Patent and its infringement thereof, or has been willfully blind as to the existence of the ’303 Patent and its infringement thereof. As one example, Google is a founding member of the Alliance for Open Media, the organization that publishes the AV1 Specification, as of the time of its inception in 2015. Google’s Head of Strategy and Partnerships for Chrome Media, Matt Frost, is currently listed as the Chair of AOM’s Steering Committee. The Alliance for Open Media’s stated goal was to create a video codec that was free of patent licensing

[36] *See, e.g.*, https://store.google.com/magazine/support?hl=en-US

obligations associated with prior video codecs.[37] For example, Google's YouTube product previously used a video codec called HEVC, and Google was motivated to avoid HEVC licensing fees by developing AV1 through the Alliance for Open Media.[38] The Alliance for Open Media, including Google, conducted a "comprehensive evaluation of the video codec patent landscape and performance of patent due diligence by world-class codec engineers and legal professionals during the development stage."[39] Upon information and belief, this "patent due diligence" either uncovered the existence of the '303 Patent and Google's infringement thereof, or should have uncovered the existence of the '303 Patent and Google's infringement thereof. Google could not have reasonably believed that the development of the AV1 video codec could not infringe any valid patent claims, including those of the '303 Patent.

125. Upon information and belief, Defendant had actual knowledge of the '303 Patent from related prior litigations accusing products with similar AV1 functionalities involving direct competitors of Defendant.

126. ACT has suffered damages as a result of Defendant's direct, indirect, and willful infringement of the '303 Patent in an amount to be proved at trial.

COUNT III
(Infringement of the '995 Patent)

127. Paragraphs 1 through 90 are incorporated by reference as if fully set forth herein.

128. ACT has not licensed or otherwise authorized Defendant to make, use, offer for sale, sell, or import any products that embody the inventions of the '995 Patent.

129. Defendant has and continues to directly infringe the '995 Patent, either literally or

[37] https://aomedia.org/press%20releases/alliance-to-deliver-next-generation-open-media-formats/
[38] https://arstechnica.com/information-technology/2015/09/microsoft-google-amazon-others-aim-for-royalty-free-video-codecs/
[39] https://aomedia.org/press%20releases/the-alliance-for-open-media-statement/

under the doctrine of equivalents, without authority and in violation of 35 U.S.C. § 271, by making, using, offering to sell, selling, and/or importing into the United States products that satisfy each and every limitation of one or more claims of the ’995 Patent. Such products include at least Google Pixel Smartphone products compliant with the AV1 and/or SVT-AV1 Standards including, but not limited to, the Pixel 6, Pixel 6 Pro, Pixel 6a, Pixel 7, Pixel 7 Pro, Pixel 7a, Pixel Fold, Pixel 8, Pixel 8 Pro, Pixel 8a, Google Chromebook laptop products, Google Pixel Slate, Google Chrome, Google Chromecast, Android/Google TV, Youtube, Google Meet, and Google Duo (the ’995 Accused Products) which include a demultiplexer configured to work on a sequence of input encoded bits to implement a process for a prescribed demultiplexing to output at least a first and a second sequence of encoded bits; a first decoder configured to acquire the first sequence of encoded bits obtained with a standard resolution at the demultiplexer to implement thereon a process for a prescribed first decoding to create a sequence of decoded pictures with a standard resolution; a first super-resolution enlarger configured to acquire the sequence of decoded pictures created with a standard resolution at the first decoder to work on the sequence of decoded pictures to implement an interpolation of pixels with a first enlargement to create a sequence of super-resolution enlarged decoded pictures with a first resolution higher than a standard resolution; a first resolution converter configured to acquire the sequence of super-resolution enlarged decoded pictures created at the first super-resolution enlarger to work on the sequence of super-resolution enlarged decoded pictures to implement a process for a prescribed resolution conversion to create a sequence of super-resolution decoded pictures with a standard resolution; a second decoder configured to acquire the second sequence of encoded bits obtained with a standard resolution at the demultiplexer as a set of decoding targets, the sequence of decoded pictures created with the standard resolution at the first decoder as a set of first reference pictures, and the sequence of

super-resolution decoded pictures created with the standard resolution at the first resolution converter as a set of second reference pictures, and select one of the set of first reference pictures and the set of second reference pictures based on reference picture selection information to implement a combination of processes for a prescribed prediction and a prescribed second decoding being a decoding with an extension of the standard resolution, to create a sequence of super-resolution pictures decoded with the standard resolution based on the set of decoding targets and the set of selected reference pictures; and a second resolution converter configured to acquire the sequence of decoded pictures with the standard resolution from the first decoder to work on the sequence of decoded pictures to implement an interpolation of pixels with the second enlargement to create a sequence of enlarged decoded pictures with a high resolution as a second resolution higher than the standard resolution, wherein the set of decoding targets, the set of first reference pictures, and the set of second reference pictures have the same value in spatial resolution.

130. For example, Defendant has and continues to directly infringe at least claim 2 of the ’995 Patent by making, using, offering to sell, selling, and/or importing into the United States the ’995 Accused Products.

131. The ’995 Accused Products include a demultiplexer configured to work on a sequence of input encoded bits to implement a process for a prescribed demultiplexing to output at least a first and a second sequence of encoded bits. AV1 and/or SVT-AV1 consist of a pipeline with either super-resolution being active or inactive for each frame. The demultiplexer generates two sequences of bits, the first sequence of bits being the I-Frames sent to a first decoder, and the second sequence of bits being P-Frames sent to a second decoder:

5.9.2. Uncompressed header syntax

```
FrameIsIntra = 1
```

Source: https://aomediacodec.github.io/av1-spec/av1-spec.pdf, Page 37-38

II. HIGH-LEVEL SYNTAX

The AV1 bitstream is packetized into open bitstream units (OBUs). An ordered sequence of OBUs is fed into the AV1 decoding process, where each OBU comprises a variable length string of bytes. An OBU contains a header and a payload. The header identifies the OBU type and specifies the payload size. Typical OBU types include the following.

1) **Sequence Header** contains information that applies to the entire sequence, e.g., sequence profile (see Section VIII) and whether to enable certain coding tools.
2) **Temporal Delimiter** indicates the frame presentation time stamp. All displayable frames following a temporal delimiter OBU will use this time stamp, until the next temporal delimiter OBU arrives. A temporal delimiter and its subsequent OBUs of the same time stamp are referred to as a temporal unit. In the context of scalable coding, the compression data associated with all representations of a frame at various spatial and fidelity resolutions will be in the same temporal unit.
3) **Frame Header** sets up the coding information for a given frame, including signaling inter or intraframe type, indicating the reference frames and signaling probability model update method.
4) **Tile Group** contains the tile data associated with a frame. Each tile can be independently decoded. The collective reconstructions form the reconstructed frame after potential loop filtering.
5) **Frame** contains the frame header and tile data. The frame OBU is largely equivalent to a frame header OBU and a tile group OBU but allows less overhead cost.
6) **Metadata** carries information, such as high dynamic range, scalability, and timecode.
7) **Tile List** contains tile data similar to a tile group OBU. However, each tile here has an additional header that indicates its reference frame index and position in the current frame. This allows the decoder to process a subset of tiles and display the corresponding part of the frame, without the need to fully decode all the tiles in the frame. Such capability is desirable for light field applications [13].

We refer to [9] for bit field definitions and more detailed consideration of high-level syntax.

132. The ’995 Accused Products include a first decoder configured to acquire the first sequence of encoded bits and decode the I-Frames received from the demultiplexer:

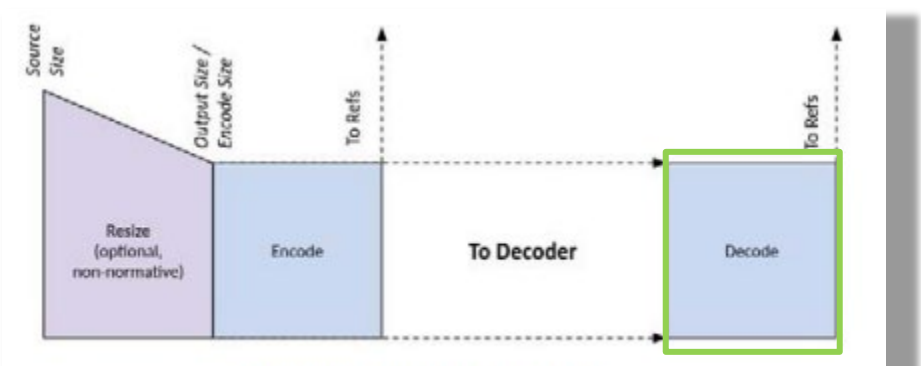
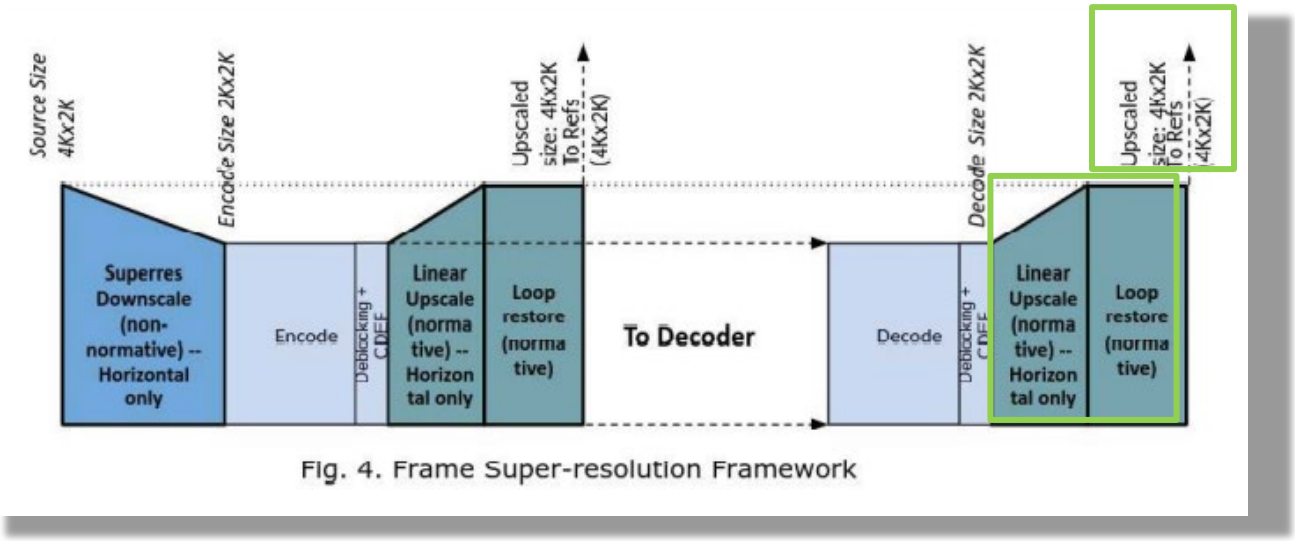


Fig. 1. Scaled prediction framework

Source: https://sci-hub.se/https://ieeexplore.ieee.org/document/8954553

133. The ’995 Accused Products include a first super-resolution enlarger configured to acquire the sequence of decoded pictures created with a standard resolution at the first decoder.

With super-resolution active, after the normal decoding process is completed, the decoded I-Frames *(i.e.*, sequence of decoded pictures created with a standard resolution at the first decoder) are further sent to the deblocking, CDEF, upscale, and loop restoration block, where the decoded pictures are enlarged and upscaled to the original resolution (*i.e.*, higher than the standard resolution). In AV1 and/or SVT-AV1, the upscaling and loop restoration operations are referred to as the super-resolve steps *(i.e.*, the first super-resolution enlarger):



Source: https://sci-hub.se/https://ieeexplore.ieee.org/document/8954553

7.16. Upscaling process

Input to this process is an array inputFrame of width FrameWidth and height FrameHeight.

The output of this process is a horizontally upscaled frame of width UpscaledWidth and height FrameHeight.

If use_superres is equal to 0, no upscaling is required and this process returns inputFrame.

Source: https://aomediacodec.github.io/av1-spec/av1-spec.pdf, Page 325

134. The ’995 Accused Products include a first resolution converter configured to acquire the sequence of super-resolution enlarged decoded pictures created at the first super-resolution enlarger to work on the sequence of super-resolution enlarged decoded pictures to

implement a process for a prescribed resolution conversion to create a sequence of super-resolution decoded pictures with a standard resolution. After the loop restoration process, the reconstructed I-Frames are added to the reference buffer list which are further used for decoding of P-Frames. The reference pictures at the decoding side are scaled according to the resolution of current P-frame which is to be decoded. Since the first super-resolution enlarger provides upscaled decoded reference pictures, the reference pictures are downscaled to match current P-Frame's resolution (frame being decoded by 2nd decoder) to be used as reference picture:

A. Scaled Prediction

In order to enable the codec to switch frame resolutions mid-stream, both AV1 and its predecessor VP9 support the ability to predict across scales in the inter prediction loop. As shown schematically in Fig. 1, this allows any frame or frames to be non-normatively downscaled or upscaled (Fig. 1 shows downscaling only) on-the-fly before encoding at a different resolution. The reconstructed frame after encoding at the reduced or increased resolution then replaces one of the reference buffer slots at that resolution. Therefore, at any point during the encoding and decoding process, any inter frame could be predicted from references that are at different resolutions, and consequently a normative mechanism to predict a block in that frame from a different resolution reference buffer needs to be defined. In principle, as long as we have defined a normative upscaler and a normative downscaler, such prediction across scales would be possible to support. However it would be more compute efficient to combine such rescaling with subpel interpolation for motion compensation, and that is what AV1 does.

Source: https://sci-hub.se/https://ieeexplore.ieee.org/document/8954553

135. The '995 Accused Products include a second decoder configured to acquire the second sequence of encoded bits obtained with a standard resolution at the demultiplexer as a set of decoding targets, the sequence of decoded pictures created with the standard resolution at the first decoder as a set of first reference pictures, and the sequence of super-resolution decoded pictures created with the standard resolution at the first resolution converter as a set of second

reference pictures, and select one of the set of first reference pictures and the set of second reference pictures based on reference picture selection information to implement a combination of processes for a prescribed prediction and a prescribed second decoding being a decoding with an extension of the standard resolution, to create a sequence of super-resolution pictures decoded with the standard resolution based on the set of decoding targets and the set of selected reference pictures. The second decoder decodes the P-Frames. When frames are decoded without super-resolution being active and being used as reference frames, the reconstructed frames are used for inter-prediction of the current frame. When super-resolution is active, AV1 and/or SVT-AV1 produce decoded frames which are references that are super-resolved and then downscaled to match the current frame resolution. The second decoder waits for the current P-Frame to be decoded as received from the demultiplexer, and when it is received, the frame can be decoded based on the relevant reference I-Frame, whether super-resolved or non-super-resolved:

A. Scaled Prediction

In order to enable the codec to switch frame resolutions mid-stream, both AV1 and its predecessor VP9 support the ability to predict across scales in the inter prediction loop. As shown schematically in Fig. 1, this allows any frame or frames to be non-normatively downscaled or upscaled (Fig. 1 shows downscaling only) on-the-fly before encoding at a different resolution. The reconstructed frame after encoding at the reduced or increased resolution then replaces one of the reference buffer slots at that resolution. Therefore, at any point during the encoding and decoding process, any inter frame could be predicted from references that are at different resolutions, and consequently a normative mechanism to predict a block in that frame from a different resolution reference buffer needs to be defined. In principle, as long as we have defined a normative upscaler and a normative downscaler, such prediction across scales would be possible to support. However it would be more compute efficient to combine such rescaling with subpel interpolation for motion compensation, and that is what AV1 does.

Source: https://sci-hub.se/https://ieeexplore.ieee.org/document/8954553

136. Since AV1 and/or SVT-AV1 allow each frame to either be normally decoded or decoded with super-resolve steps, the reference picture buffer set consists of both non-super-resolved and super-resolved reference pictures (reconstructed frames). For the second decoder to decode the current frame, the reference frame is selected based on the reference index. The reference index, which indicates whether a super-resolved or non-super-resolved reconstructed frame is selected, is the reference picture selection information that is sent in the encoded bitstream.

ref_frame_idx[i] specifies which reference frames are used by inter frames. It is a requirement of bitstream conformance that RefValid[ref_frame_idx[i]] is equal to 1, and that the selected reference frames match the current frame in bit depth, profile, chroma subsampling, and color space.

Note: Syntax elements indicate a reference (such as LAST_FRAME, ALTREF_FRAME). These references are looked up in the ref_frame_idx array to find which reference frame should be used during inter prediction. There is no requirement that the values in ref_frame_idx should be distinct.

Source: https://aomediacodec.github.io/av1-spec/av1-spec.pdf, Page 327

137. The ’995 Accused Products include a second resolution converter configured to acquire the sequence of decoded pictures with the standard resolution from the first decoder to work on the sequence of decoded pictures to implement an interpolation of pixels with the second enlargement to create a sequence of enlarged decoded pictures with a high resolution as a second resolution higher than the standard resolution, wherein the set of decoding targets, the set of first reference pictures, and the set of second reference pictures have the same value in spatial resolution. In AV1 and/or SVT-AV1, the output of the 1st decoder (when super-resolution is not active), the decoded frames (reconstructed references) can also be upscaled. AV1 and/or SVT-AV1 use different 8-tap filter coefficients that can be used for upscaling of the decoded frame.

```
const int16_t av1_resize_filter_normative[(
    1 << RS_SUBPEL_BITS)][UPSCALE_NORMATIVE_TAPS] = {
#if UPSCALE_NORMATIVE_TAPS == 8
  { 0, 0, 0, 128, 0, 0, 0, 0 },        { 0, 0, -1, 128, 2, -1, 0, 0 },
  { 0, 1, -3, 127, 4, -2, 1, 0 },      { 0, 1, -4, 127, 6, -3, 1, 0 },
  { 0, 2, -6, 126, 8, -3, 1, 0 },      { 0, 2, -7, 125, 11, -4, 1, 0 },
  { -1, 2, -8, 125, 13, -5, 2, 0 },    { -1, 3, -9, 124, 15, -6, 2, 0 },
```

Source: https://aomedia.googlesource.com/aom/+/refs/heads/main/av1/common/resize.c

After the reference pictures are selected from the first and second set of reference pictures, the reference pictures are upscaled or downscaled to match to resolution of the encoding targets:

Fig. 2 shows a scenario where a 4x4 block from the source needs to be predicted from a different resolution reference buffer. The motion vectors transmitted in the bitstream are always at the source resolution. So they are first scaled up or down based on the resolution ratio between the reference and source, and the corresponding source block pixels projected on the reference grid can then be obtained, as shown on the right of Fig. 2. Note that the relative sub-pixel positions horizontally (vertically) on the reference grid are the same in each row (column). Hence, the interpolation for scaled prediction can be implemented simply as separable filtering in each dimension using a suitable starting sub-pixel offset and a sub-pixel step between pixels.

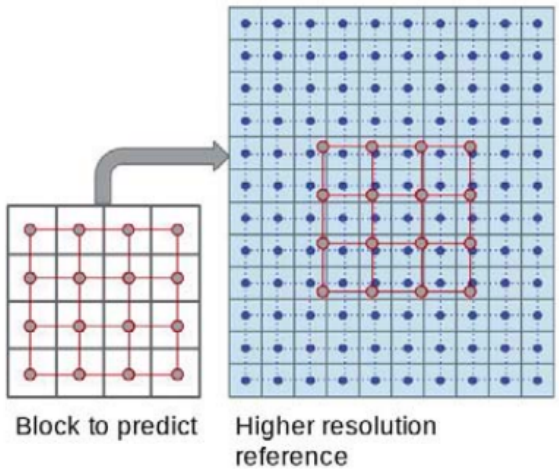


Fig. 2. Predicting a block from different (higher shown) resolution reference

Source: https://sci-hub.se/https://ieeexplore.ieee.org/document/8954553, Page 2

138. Google has indirectly infringed and continues to indirectly infringe one or more claims of the ’995 Patent, as provided by 35 U.S.C. § 271(b), by inducing infringement by others, such as Google’s customers and end-users, in this Judicial District and elsewhere in the United States. For example, Google’s customers and end-users directly infringe, either literally or under the doctrine of equivalents, through their use of the inventions claimed in the ’995 Patent. Google induces this direct infringement through its affirmative acts of manufacturing, selling, distributing, and/or otherwise making available the ’995 Accused Products, and providing instructions, documentation, and other information to customers and end-users suggesting that they use the ’995 Accused Products in an infringing manner, including technical support, marketing, product manuals, advertisements, and online documentation.[40] Because of Google’s inducement, Google’s customers and end-users use the ’995 Accused Products in a way Google intends and they directly infringe the ’995 Patent. Google performs these affirmative acts with knowledge of the ’995 Patent and with the intent, or willful blindness, that the induced acts directly infringe the ’995 Patent.

139. Google has indirectly infringed and continues to indirectly infringe one or more claims of the ’995 Patent, as provided by 35 U.S.C. § 271(c), by contributing to direct infringement by others, such as customers and end-users, in this Judicial District and elsewhere in the United States. Google’s affirmative acts of selling and offering to sell the ’995 Accused Products in this Judicial District and elsewhere in the United States and causing the ’995 Accused Products to be manufactured, used, sold, and offered for sale contribute to others’ use and manufacture of the ’995 Accused Products, such that the ’995 Patent is directly infringed by others.[41] The accused

---

[40] *See, e.g.*, https://guidebooks.google.com/home; https://pixel.withgoogle.com/?hl=en&country=US; https://pixel.withgoogle.com/support/Pixel?hl=en&country=US&dark=0; https://www.youtube.com/@Googlehelp/playlists

[41] *See, e.g.*, https://store.google.com/magazine/support?hl=en-US

components within the ’995 Accused Products including, but not limited to, software manufactured by Google, are material to the invention of the ’995 Patent, are not staple articles or commodities of commerce, have no substantial non-infringing uses, and are known by Google to be especially made or adapted for use in the infringement of the ’995 Patent. Google performs these affirmative acts with knowledge of the ’995 Patent and with intent, or willful blindness, that they cause the direct infringement of the ’995 Patent.

140. Google’s infringement of the ’995 Patent is and has been willful. Google was on notice of the existence of the ’995 Patent and its infringement thereof, or has been willfully blind as to the existence of the ’995 Patent and its infringement thereof. As one example, Google is a founding member of the Alliance for Open Media, the organization that publishes the AV1 Specification, as of the time of its inception in 2015. Google’s Head of Strategy and Partnerships for Chrome Media, Matt Frost, is currently listed as the Chair of AOM’s Steering Committee. The Alliance for Open Media’s stated goal was to create a video codec that was free of patent licensing obligations associated with prior video codecs.[42] For example, Google’s YouTube product previously used a video codec called HEVC, and Google was motivated to avoid HEVC licensing fees by developing AV1 through the Alliance for Open Media.[43] The Alliance for Open Media, including Google, conducted a “comprehensive evaluation of the video codec patent landscape and performance of patent due diligence by world-class codec engineers and legal professionals during the development stage.”[44] Upon information and belief, this “patent due diligence” either uncovered the existence of the ’995 Patent and Google’s infringement thereof, or should have

[42] https://aomedia.org/press%20releases/alliance-to-deliver-next-generation-open-media-formats/
[43] https://arstechnica.com/information-technology/2015/09/microsoft-google-amazon-others-aim-for-royalty-free-video-codecs/
[44] https://aomedia.org/press%20releases/the-alliance-for-open-media-statement/

uncovered the existence of the ’995 Patent and Google’s infringement thereof. Google could not have reasonably believed that the development of the AV1 video codec could not infringe any valid patent claims, including those of the ’995 Patent.

141. Upon information and belief, Defendant had actual knowledge of the ’995 Patent from related prior litigations accusing products with similar AV1 functionalities involving direct competitors of Defendant.

142. Defendant has been on actual notice of the ’995 Patent and Defendant’s infringement thereof at least as of June 3, 2013, when it was cited during prosecution of U.S. Patent No. 8,635,357, entitled “Dynamic selection of parameter sets for transcoding media data,” and currently assigned to Google LLC. Specifically, during prosecution, the Examiner cited to U.S. Patent Publication No. 2011/0075734 A1, which is the published patent application number of the ’448 Patent, of which the ’995 patent application is a divisional patent application.

143. ACT has suffered damages as a result of Defendant’s direct, indirect, and willful infringement of the ’995 Patent in an amount to be proved at trial.

COUNT IV
(Infringement of the ’448 Patent)

144. Paragraphs 1 through 90 are incorporated by reference as if fully set forth herein.

145. ACT has not licensed or otherwise authorized Defendant to make, use, offer for sale, sell, or import any products that embody the inventions of the ’448 Patent.

146. Defendant has and continues to directly infringe the ’448 Patent, either literally or under the doctrine of equivalents, without authority and in violation of 35 U.S.C. § 271, by making, using, offering to sell, selling, and/or importing into the United States products that satisfy each and every limitation of one or more claims of the ’448 Patent. Such products include at least Google Pixel Smartphone products including, but not limited to, the Pixel 6, Pixel 6 Pro, Pixel 6a,

Pixel 7, Pixel 7 Pro, Pixel 7a, Pixel Fold, Pixel 8, Pixel 8 Pro, and Pixel 8a, Google Chromebook laptop products, Google Pixel Slate, Google Chrome, Google Chromecast, Android/Google TV, Youtube, Google Meet, and Google Duo (the ’448 Accused Products), each of which is a moving picture encoding system that makes an encoding of a sequence of moving pictures with a resolution higher than a standard resolution using moving pictures contents which include a sequence of moving pictures with the standard resolution and do not include a sequence of moving pictures with a resolution higher than the standard resolution, the moving picture encoding system comprising: a first encoder configured to work on a sequence of moving pictures with a standard resolution to implement a first combination of processes for an encoding and a decoding to create a first sequence of encoded bits and a set of decoded pictures with the standard resolution; a first super-resolution enlarger configured to work on the sequence of moving pictures with the standard resolution to implement a process for a first super-resolution enlargement to create a set of super-resolution enlarged pictures with a resolution higher than the standard resolution; a second super-resolution enlarger configured to acquire the set of decoded pictures from the first encoder to implement thereon a process for a second super-resolution enlargement to create a set of super-resolution enlarged decoded pictures with a resolution higher than the standard resolution; a third resolution converter configured to acquire the set of decoded pictures from the first encoder to implement thereon a process for a third resolution conversion to create a set of resolution converted enlarged decoded pictures with a resolution higher than the standard resolution; and a third encoder configured to have the set of super-resolution enlarged pictures from the first super-resolution enlarger as a set of encoding target pictures, employing the set of super-resolution enlarged decoded pictures from the second super-resolution enlarger and the set of resolution converted enlarged decoded pictures from the third resolution converter as sets of reference pictures, to

implement thereon a third combination of processes for a prediction and an encoding to create a third sequence of encoded bits, wherein a spatial resolution of the set of super-resolution enlarged pictures, that of the set of super-resolution enlarged decoded pictures, and that of the set of resolution converted enlarged decoded pictures are made equal; and wherein the third encoder controls a selection of a set of reference pictures and creates a set of data on the selection of the set of reference pictures to identify a selected set of reference pictures during the process for prediction of the third combination.

147. For example, Defendant has and continues to directly infringe at least claim 1 of the ’448 Patent by making, using, offering to sell, selling, and/or importing into the United States the ’448 Accused Products.

148. Each of the ’448 Accused Products is a moving picture encoding system that makes an encoding of a sequence of moving pictures with a resolution higher than a standard resolution using moving pictures contents which include a sequence of moving pictures with the standard resolution and do not include a sequence of moving pictures with a resolution higher than the standard resolution, the moving picture encoding system. For example, the libaom-av1 encoder, an implementation of the AV1 encoder specification developed and used by the ’448 Accused Products, allows the input video pictures at a higher resolution to be coded at a low resolution. Libaom-av1 allows encoding with different sizes of reference frames (i.e., upscaled or downscaled frames). The encoding pipeline in libaom-av1 consists of different open-loop and closed-loop processes.

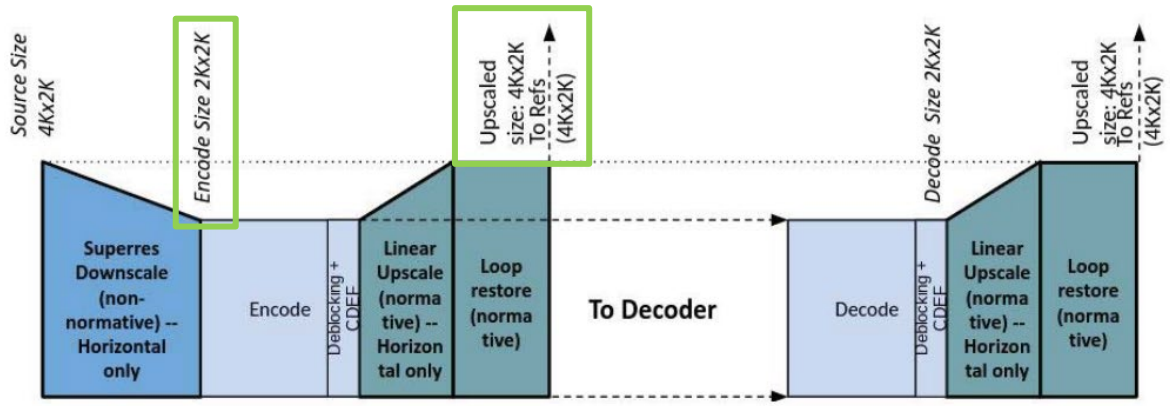
***Abstract*— AV1 is a recently standardized royalty-free video codec from the industry consortium Alliance for Open Media. One of the most innovative coding tools supported in AV1 is an in-loop frame super-resolution mode, that allows an encoder to code any frame at a horizontally reduced spatial resolution by one of several levels, followed by upsampling and super-resolving to full resolution, before replacing reference buffers. This mode is partly enabled by a feature in AV1 that natively allows the motion compensated prediction loop to operate across scales between a coded frame and the available references, thereby allowing on-the-fly resolution change mid-stream within a sequence. For the actual super-resolving process a normative upscaler is followed by an in-loop restoration tool that recovers some of the high frequency information lost in the downsampling process. On-the-fly resolution change capability in conjunction with the frame-superresolution mode in AV1 opens up a new dimension for codec bitrate and quality optimization that has not been possible to explore in any prior standardized video codec with (soon expected) decoding hardware support. This paper provides an overview of how the relevant tools work in AV1, but unlocking them with intelligent encoder decisions to extract real-world benefit, is largely left as future work.**

Source: IEEE, "In-loop Frame Super-resolution in AV1," at 1 (available at https://ieeexplore.ieee.org/document/8954553).

3.7.4 Frame super-resolution

To improve the perceptual quality of decoded pictures, a super-resolution process can be applied at low bitrates. The super-resolution process can be performed on a per-frame basis. First, at the encoder side, the source video is downscaled as a nonnormative procedure. Second, the downscaled video is encoded, followed by deblocking and CDEF filtering processes. Third, a linear upscaling process is applied as a normative procedure to convert the encoded video back to its original resolution. Finally, an LR filter is applied to further recover some lost details of the upscaled picture. The last two steps make up the super-resolving process; the deblocking and CDEF filters are applied at lower spatial resolution on the decoder side, after which the frames are passed through the super-resolution process. To reduce complexity regarding the use of line buffers in hardware implementation, the upscaling and downscaling processes are applied to the horizontal dimension only.

Source: https://aomedia.org/docs/AV1_ToolDescription_v11-clean.pdf, 36.



Source: IEEE, "In-loop Frame Super-resolution in AV1," at 3 (available at https://ieeexplore.ieee.org/document/8954553).

resolution. Therefore, at any point during the encoding and decoding process, any inter frame could be predicted from references that are at different resolutions, and consequently a normative mechanism to predict a block in that frame from a different resolution reference buffer needs to be defined. In principle, as long as we have defined a normative upscaler and a normative downscaler, such prediction across scales would be possible to support. However it would be

Source: IEEE, "In-loop Frame Super-resolution in AV1," at 4 (available at https://ieeexplore.ieee.org/document/8954553).

Videoconferencing brings us together, even when we're thousands of miles apart. Whether it's catching up with your best friend, or having a quick meeting with a coworker, video calls have become an integral part of our lives.

State of the art video codecs such as AV1 can dramatically improve video call quality, while using less bandwidth to transmit higher quality video. We're now seeing AV1 hardware encoders in desktops and laptops, but what about users who have older devices?

The Chrome Open Media team has worked for the last 18 months to make the AV1 software encoder (libaom) much faster, unlocking better quality AV1 video calling on all devices. Chrome 113 includes libaom v3.6.0, which greatly improves real-time communications performance—both speed and quality—for speeds 6 through 10. In particular we added a "Speed 10" setting for devices with limited CPU capabilities.

Source: https://developer.chrome.com/blog/av1/, at 1.

Google Releases libaom 3.7 AV1 Encoder With More Optimizations

Written by Michael Larabel in Multimedia on 1 September 2023 at 05:52 AM EDT. 13 Comments

Following the releases this week of Intel's SVT-AV1 1.7 encoder and the libavif 1.0 AV1 Image File Format release, Google engineers are out with libaom 3.7 as the newest feature release to that AV1 encode library.

The libaom 3.7 release focuses on new codec interfaces, compression efficiency and perceptual improvements, and speed-ups and memory optimizations. There is also a new speed (--cpu-used 11) and other enhancements.

Source: https://www.phoronix.com/news/Google-libaom-3.7-AV1.

149. The ’448 Accused Products include a first encoder configured to work on a sequence of moving pictures with a standard resolution to implement a first combination of processes for an encoding and a decoding to create a first sequence of encoded bits and a set of decoded pictures with the standard resolution. The libaom-av1 implementation of AV1 encoder, used by the ’448 Accused Products, includes a first encoder that encodes the images applied at the input. When super-resolution mode is disabled (i.e., in a first pass), the encoder takes the input images at the original resolution and produces encoded images at the same resolution, such as in open loop I-frame encoding:

Videoconferencing brings us together, even when we're thousands of miles apart. Whether it's catching up with your best friend, or having a quick meeting with a coworker, video calls have become an integral part of our lives.

State of the art video codecs such as AV1 can dramatically improve video call quality, while using less bandwidth to transmit higher quality video. We're now seeing AV1 hardware encoders in desktops and laptops, but what about users who have older devices?

The Chrome Open Media team has worked for the last 18 months to make the AV1 software encoder (libaom) much faster, unlocking better quality AV1 video calling on all devices. Chrome 113 includes libaom v3.6.0, which greatly improves real-time communications performance—both speed and quality—for speeds 6 through 10. In particular we added a "Speed 10" setting for devices with limited CPU capabilities.

Source: https://developer.chrome.com/blog/av1/.

```
static int denoise_and_encode(AV1_COMP *const cpi, uint8_t *const dest,
                              EncodeFrameInput *const frame_input,
                              EncodeFrameParams *const frame_params,
                              EncodeFrameResults *const frame_results) {
  const AV1EncoderConfig *const oxcf = &cpi->oxcf;
  AV1_COMMON *const cm = &cpi->common;
```

Source: https://github.com/ultrawide/libaom/blob/master/av1/encoder/encodemv.h

```
if (!encode_show_existing_frame(cm)) {
  // Refresh fb_of_context_type[]: see encoder.h for explanation
  if (cm->current_frame.frame_type == KEY_FRAME) {
    // All ref frames are refreshed, pick one that will live long enough
    fb_of_context_type[current_frame_ref_type] = 0;
  } else {
    // If more than one frame is refreshed, it doesn't matter which one we
    // pick so pick the first.  LST sometimes doesn't refresh any: this is ok

    for (int i = 0; i < REF_FRAMES; i++) {
      if (cm->current_frame.refresh_frame_flags & (1 << i)) {
        fb_of_context_type[current_frame_ref_type] = i;
        break;
      }
    }
  }
}
}
```

```
void av1_encode_mv(AV1_COMP *cpi, aom_writer *w, const MV *mv, const MV *ref,
                   nmv_context *mvctx, int usehp);
```

Source: https://github.com/ultrawide/libaom/blob/master/av1/encoder/encodemv.h

150. The ’448 Accused Products include a first super-resolution enlarger configured to work on the sequence of moving pictures with the standard resolution to implement a process for a first super-resolution enlargement to create a set of super-resolution enlarged pictures with a resolution higher than the standard resolution. After subsequent processing steps in the AV1 encoding process (e.g., deblocking, CDEF), downscaled or upscaled input images and reference

images are upscaled and loop restored for super resolution functionality. Thus, the upscaled and encoded pictures generated based on the downscaled input and reference images are super-resolution enlarged pictures, which have a resolution higher than the input images. The component implementing upscaling of the input images for the above processing step acts as the first resolution enlarger.

A. Scaled Prediction

In order to enable the codec to switch frame resolutions mid-stream, both AV1 and its predecessor VP9 support the ability to predict across scales in the inter prediction loop. As shown schematically in Fig. 1, this allows any frame or frames to be non-normatively downscaled or upscaled (Fig. 1 shows downscaling only) on-the-fly before encoding at a different resolution. The reconstructed frame after encoding at the reduced or increased resolution then replaces one of the reference buffer slots at that resolution. Therefore, at any point during the encoding and decoding process, any inter frame could be predicted from references that are at different resolutions, and consequently a normative mechanism to predict a block in that frame from a different resolution reference buffer needs to be defined. In principle, as long as we have defined a normative upscaler and a normative downscaler, such prediction across scales would be possible to support. However it would be more compute efficient to combine such rescaling with subpel interpolation for motion compensation, and that is what AV1 does.

Source: IEEE, "In-loop Frame Super-resolution in AV1," at 2 (available at https://ieeexplore.ieee.org/document/8954553).

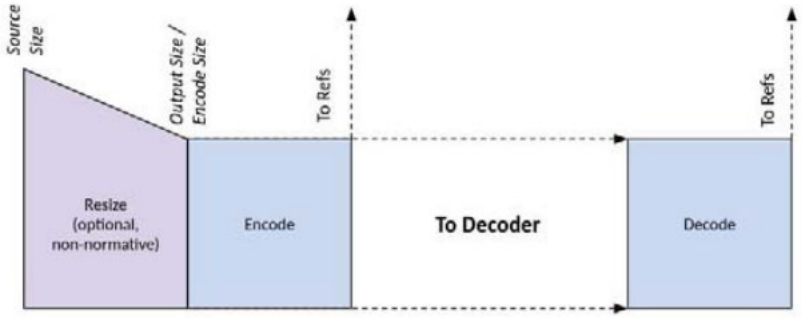


Fig. 1. Scaled prediction framework

Fig: 11, IEEE, "In-loop Frame Super-resolution in AV1," at 2 (available at https://ieeexplore.ieee.org/document/8954553).

```
// Encode frames.
while (aom_img_read(raw, infile) && frame_count < limit) {
  ++frame_count;
  encode_frame(&codec, raw, frame_count, 1, 0, writer);
}
```

Source: https://github.com/ultrawide/libaom/blob/master/examples/twopass_encoder.c

```
void av1_encode_mv(AV1_COMP *cpi, aom_writer *w, const MV *mv, const MV *ref,
                   nmv_context *mvctx, int usehp);
```

Source: https://github.com/ultrawide/libaom/blob/master/av1/encoder/encodemv.h

```
static int encode_frame(aom_codec_ctx_t *ctx, const aom_image_t *img,
                        aom_codec_pts_t pts, unsigned int duration,
                        aom_enc_frame_flags_t flags, AvxVideoWriter *writer) {
  int got_pkts = 0;
  aom_codec_iter_t iter = NULL;
  const aom_codec_cx_pkt_t *pkt = NULL;
  const aom_codec_err_t res = aom_codec_encode(ctx, img, pts, duration, flags);
  if (res != AOM_CODEC_OK) die_codec(ctx, "Failed to encode frame.");

  while ((pkt = aom_codec_get_cx_data(ctx, &iter)) != NULL) {
    got_pkts = 1;
    if (pkt->kind == AOM_CODEC_CX_FRAME_PKT) {
      const int keyframe = (pkt->data.frame.flags & AOM_FRAME_IS_KEY) != 0;

      if (!aom_video_writer_write_frame(writer, pkt->data.frame.buf,
                                        pkt->data.frame.sz,
                                        pkt->data.frame.pts))
        die_codec(ctx, "Failed to write compressed frame.");
      printf(keyframe ? "K" : ".");
      fflush(stdout);
    }
  }
```

Source: https://github.com/ultrawide/libaom/blob/master/examples/twopass_encoder.c

3.7.4 Frame super-resolution

To improve the perceptual quality of decoded pictures, a super-resolution process can be applied at low bitrates. The super-resolution process can be performed on a per-frame basis. First, at the encoder side, the source video is downscaled as a nonnormative procedure. Second, the downscaled video is encoded, followed by deblocking and CDEF filtering processes. Third, a linear upscaling process is applied as a normative procedure to convert the encoded video back to its original resolution. Finally, an LR filter is applied to further recover some lost details of the upscaled picture. The last two steps make up the super-resolving process; the deblocking and

Source: https://aomedia.org/docs/AV1_ToolDescription_v11-clean.pdf, 36

```
// Encode frames.
while (aom_img_read(&raw0, infile0)) {
  int flags = 0;

  // configure and encode base layer

  if (keyframe_interval > 0 && frames_encoded % keyframe_interval == 0)
    flags |= AOM_EFLAG_FORCE_KF;
  else
    // use previous base layer (LAST) as sole reference
    // save this frame as LAST to be used as reference by enhanmcent layer
    // and next base layer
    flags |= AOM_EFLAG_NO_REF_LAST2 | AOM_EFLAG_NO_REF_LAST3 |
             AOM_EFLAG_NO_REF_GF | AOM_EFLAG_NO_REF_ARF |
             AOM_EFLAG_NO_REF_BWD | AOM_EFLAG_NO_REF_ARF2 |
             AOM_EFLAG_NO_UPD_GF | AOM_EFLAG_NO_UPD_ARF |
             AOM_EFLAG_NO_UPD_ENTROPY;
  cfg.g_w = info.frame_width;
  cfg.g_h = info.frame_height;
  if (aom_codec_enc_config_set(&codec, &cfg))
    die_codec(&codec, "Failed to set enc cfg for layer 0");
  if (aom_codec_control(&codec, AOME_SET_SPATIAL_LAYER_ID, 0))
    die_codec(&codec, "Failed to set layer id to 0");
  if (aom_codec_control(&codec, AOME_SET_CQ_LEVEL, 62))
    die_codec(&codec, "Failed to set cq level");
  encode_frame(&codec, &raw0, frame_count++, flags, outfile);
```

Source: https://github.com/ultrawide/libaom/blob/master/examples/twopass_encoder.c

151. The ’448 Accused Products include a second super-resolution enlarger configured to acquire the set of decoded pictures from the first encoder to implement thereon a process for a second super-resolution enlargement to create a set of super-resolution enlarged decoded pictures with a resolution higher than the standard resolution. While in-loop processing for super-resolution

functionality is enabled, the libaom-av1 encoder uses the downscaled/upscaled reference images (i.e., reconstructed reference images) and input images. After subsequent processing steps (e.g., deblocking, CDEF) these images are upscaled and loop restored for super resolution. Thus, the upscaled and encoded reference pictures generated based on the downscaled/upscaled input images for encoding of the next set of input images have a resolution higher than the input images. The component implementing the above processing step acts as the second resolution enlarger.

> super-resolution mode in the bit-stream syntax. Specifically, if the super-resolution mode is enabled for a frame, the frame is non-normatively downsampled to a lower resolution and coded at lower resolution using higher resolution references in the motion compensation loop, followed by normative upsampling and super-resolving to generate a full resolution output frame that is displayed and that updates a reference frame store slot for use in subsequent frames. Note also that high-performant CNN based and/or

Source: IEEE, "In-loop Frame Super-resolution in AV1," at 4 (available at https://ieeexplore.ieee.org/document/8954553).

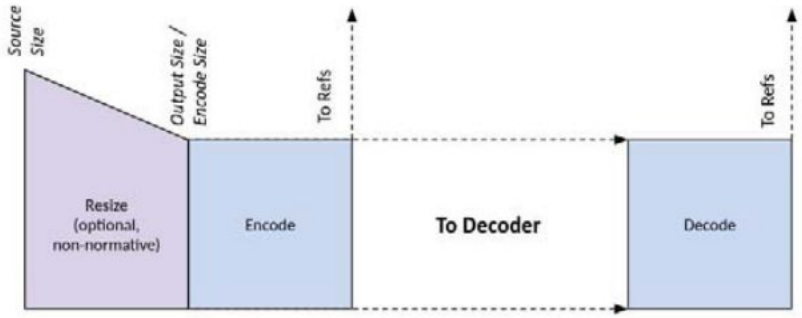
> **3.7.4 Frame super-resolution**
>
> To improve the perceptual quality of decoded pictures, a super-resolution process can be applied at low bitrates. The super-resolution process can be performed on a per-frame basis. First, at the encoder side, the source video is downscaled as a nonnormative procedure. Second, the downscaled video is encoded, followed by deblocking and CDEF filtering processes. Third, a linear upscaling process is applied as a normative procedure to convert the encoded video back to its original resolution. Finally, an LR filter is applied to further recover some lost details of the upscaled picture. The last two steps make up the super-resolving process; the deblocking and

Source: https://aomedia.org/docs/AV1_ToolDescription_v11-clean.pdf, at 36

A. Scaled Prediction

In order to enable the codec to switch frame resolutions mid-stream, both AV1 and its predecessor VP9 support the ability to predict across scales in the inter prediction loop. As shown schematically in Fig. 1, this allows any frame or frames to be non-normatively downscaled or upscaled (Fig. 1 shows downscaling only) on-the-fly before encoding at a different resolution. The reconstructed frame after encoding at the reduced or increased resolution then replaces one of the reference buffer slots at that resolution. Therefore, at any point during the encoding and decoding process, any inter frame could be predicted from references that are at different resolutions, and consequently a normative mechanism to predict a block in that frame from a different resolution reference buffer needs to be defined. In principle, as long as we have defined a normative upscaler and a normative downscaler, such prediction across scales would be possible to support. However it would be more compute efficient to combine such rescaling with subpel interpolation for motion compensation, and that is what AV1 does.

Source: IEEE, "In-loop Frame Super-resolution in AV1," at 2 (available at https://ieeexplore.ieee.org/document/8954553).



Source: IEEE, "In-loop Frame Super-resolution in AV1," at 2 (available at https://ieeexplore.ieee.org/document/8954553)

```
// Encode frames.
while (aom_img_read(raw, infile) && frame_count < limit) {
  ++frame_count;
  encode_frame(&codec, raw, frame_count, 1, 0, writer);
}
```

Source: https://github.com/ultrawide/libaom/blob/master/examples/twopass_encoder.c

```
void av1_encode_mv(AV1_COMP *cpi, aom_writer *w, const MV *mv, const MV *ref,
                   nmv_context *mvctx, int usehp);
```

Source: https://github.com/ultrawide/libaom/blob/master/av1/encoder/encodemv.h

```
static int encode_frame(aom_codec_ctx_t *ctx, const aom_image_t *img,
                        aom_codec_pts_t pts, unsigned int duration,
                        aom_enc_frame_flags_t flags, AvxVideoWriter *writer) {
  int got_pkts = 0;
  aom_codec_iter_t iter = NULL;
  const aom_codec_cx_pkt_t *pkt = NULL;
  const aom_codec_err_t res = aom_codec_encode(ctx, img, pts, duration, flags);
  if (res != AOM_CODEC_OK) die_codec(ctx, "Failed to encode frame.");

  while ((pkt = aom_codec_get_cx_data(ctx, &iter)) != NULL) {
    got_pkts = 1;
    if (pkt->kind == AOM_CODEC_CX_FRAME_PKT) {
      const int keyframe = (pkt->data.frame.flags & AOM_FRAME_IS_KEY) != 0;

      if (!aom_video_writer_write_frame(writer, pkt->data.frame.buf,
                                        pkt->data.frame.sz,
                                        pkt->data.frame.pts))
        die_codec(ctx, "Failed to write compressed frame.");
      printf(keyframe ? "K" : ".");
      fflush(stdout);
    }
```

Source: https://github.com/ultrawide/libaom/blob/master/examples/twopass_encoder.c

```
// Encode frames.
while (aom_img_read(&raw, infile)) {
  int flags = 0;
  if (keyframe_interval > 0 && frame_count % keyframe_interval == 0)
    flags |= AOM_EFLAG_FORCE_KF;
  encode_frame(&codec, &raw, frame_count++, flags, writer);
  frames_encoded++;
  if (max_frames > 0 && frames_encoded >= max_frames) break;
}

// Flush encoder.
while (encode_frame(&codec, NULL, -1, 0, writer)) continue;

printf("\n");
fclose(infile);
printf("Processed %d frames.\n", frame_count);

aom_img_free(&raw);
if (aom_codec_destroy(&codec)) die_codec(&codec, "Failed to destroy codec.");

aom_video_writer_close(writer);

return EXIT_SUCCESS;
}
```

Source: https://github.com/ultrawide/libaom/blob/master/examples/simple_encoder.c

152. The ’448 Accused Products include a third resolution converter configured to acquire the set of decoded pictures from the first encoder to implement thereon a process for a third resolution conversion to create a set of resolution converted enlarged decoded pictures with a resolution higher than the standard resolution. For example, during the encoding of I-frames, while in loop proceeding for super-resolution functionality is enabled, the libaom-av1 encoder uses the upscaled reference images (i.e., reconstructed reference images) and downscaled input images, since these downscaled input images are at 2K*2K size and encoded using upscaled reconstructed reference images and output images are produced at 4K*2K size, the component implementing the above processing step acts as the 3rd resolution converter. These images are super-resolution converted enlarged pictures, which have a resolution higher than the input images.

```
// Encode frames.
while (aom_img_read(raw, infile) && frame_count < limit) {
  ++frame_count;
  encode_frame(&codec, raw, frame_count, 1, 0, writer);
}
```

Source: https://github.com/ultrawide/libaom/blob/master/examples/twopass_encoder.c

```
void av1_encode_mv(AV1_COMP *cpi, aom_writer *w, const MV *mv, const MV *ref,
                   nmv_context *mvctx, int usehp);
```

Source: https://github.com/ultrawide/libaom/blob/master/av1/encoder/encodemv.h

```
static int encode_frame(aom_codec_ctx_t *ctx, const aom_image_t *img,
                        aom_codec_pts_t pts, unsigned int duration,
                        aom_enc_frame_flags_t flags, AvxVideoWriter *writer) {
  int got_pkts = 0;
  aom_codec_iter_t iter = NULL;
  const aom_codec_cx_pkt_t *pkt = NULL;
  const aom_codec_err_t res = aom_codec_encode(ctx, img, pts, duration, flags);
  if (res != AOM_CODEC_OK) die_codec(ctx, "Failed to encode frame.");

  while ((pkt = aom_codec_get_cx_data(ctx, &iter)) != NULL) {
    got_pkts = 1;
    if (pkt->kind == AOM_CODEC_CX_FRAME_PKT) {
      const int keyframe = (pkt->data.frame.flags & AOM_FRAME_IS_KEY) != 0;

      if (!aom_video_writer_write_frame(writer, pkt->data.frame.buf,
                                        pkt->data.frame.sz,
                                        pkt->data.frame.pts))
        die_codec(ctx, "Failed to write compressed frame.");
      printf(keyframe ? "K" : ".");
      fflush(stdout);
    }
```

Source: https://github.com/ultrawide/libaom/blob/master/examples/twopass_encoder.c

```
  // Encode frames.
  while (aom_img_read(&raw, infile)) {
    int flags = 0;
    if (keyframe_interval > 0 && frame_count % keyframe_interval == 0)
      flags |= AOM_EFLAG_FORCE_KF;
    encode_frame(&codec, &raw, frame_count++, flags, writer);
    frames_encoded++;
    if (max_frames > 0 && frames_encoded >= max_frames) break;
  }

  // Flush encoder.
  while (encode_frame(&codec, NULL, -1, 0, writer)) continue;

  printf("\n");
  fclose(infile);
  printf("Processed %d frames.\n", frame_count);

  aom_img_free(&raw);
  if (aom_codec_destroy(&codec)) die_codec(&codec, "Failed to destroy codec.");

  aom_video_writer_close(writer);

  return EXIT_SUCCESS;
}
```

Source: https://github.com/ultrawide/libaom/blob/master/examples/simple_encoder.c

3.7.4 Frame super-resolution

To improve the perceptual quality of decoded pictures, a super-resolution process can be applied at low bitrates. The super-resolution process can be performed on a per-frame basis. First, at the encoder side, the source video is downscaled as a nonnormative procedure. Second, the downscaled video is encoded, followed by deblocking and CDEF filtering processes. Third, a linear upscaling process is applied as a normative procedure to convert the encoded video back to its original resolution. Finally, an LR filter is applied to further recover some lost details of the upscaled picture. The last two steps make up the super-resolving process; the deblocking and

Source: https://aomedia.org/docs/AV1_ToolDescription_v11-clean.pdf

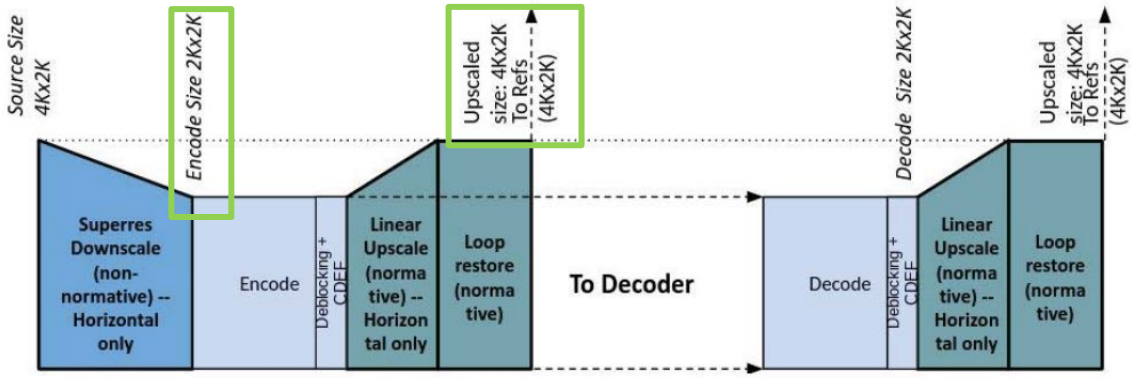


Fig. 4. Frame Super-resolution Framework

Source: IEEE, "In-loop Frame Super-resolution in AV1," at 4 (available at https://ieeexplore.ieee.org/document/8954553)

super-resolution mode in the bit-stream syntax. Specifically, if the super-resolution mode is enabled for a frame, the frame is non-normatively downsampled to a lower resolution and coded at lower resolution using higher resolution references in the motion compensation loop, followed by normative upsampling and super-resolving to generate a full resolution output frame that is displayed and that updates a reference frame store slot for use in subsequent frames. Note also that high-performant CNN based and/or

Source: IEEE, "In-loop Frame Super-resolution in AV1," at 4 (available at https://ieeexplore.ieee.org/document/8954553)

3.7.4 Frame super-resolution

To improve the perceptual quality of decoded pictures, a super-resolution process can be applied at low bitrates. The super-resolution process can be performed on a per-frame basis. First, at the encoder side, the source video is downscaled as a nonnormative procedure. Second, the downscaled video is encoded, followed by deblocking and CDEF filtering processes. Third, a linear upscaling process is applied as a normative procedure to convert the encoded video back to its original resolution. Finally, an LR filter is applied to further recover some lost details of the upscaled picture. The last two steps make up the super-resolving process; the deblocking and

Source: https://aomedia.org/docs/AV1_ToolDescription_v11-clean.pdf, 36

A. Scaled Prediction

In order to enable the codec to switch frame resolutions mid-stream, both AV1 and its predecessor VP9 support the ability to predict across scales in the inter prediction loop. As shown schematically in Fig. 1, this allows any frame or frames to be non-normatively downscaled or upscaled (Fig. 1 shows downscaling only) on-the-fly before encoding at a different resolution. The reconstructed frame after encoding at the reduced or increased resolution then replaces one of the reference buffer slots at that resolution. Therefore, at any point during the encoding and decoding process, any inter frame could be predicted from references that are at different resolutions, and consequently a normative mechanism to predict a block in that frame from a different resolution reference buffer needs to be defined. In principle, as long as we have defined a normative upscaler and a normative downscaler, such prediction across scales would be possible to support. However it would be more compute efficient to combine such rescaling with subpel interpolation for motion compensation, and that is what AV1 does.

Fig: 33, Source: IEEE, "In-loop Frame Super-resolution in AV1," at 2 (available at https://ieeexplore.ieee.org/document/8954553)

153. The '448 Accused Products include a third encoder configured to have the set of super-resolution enlarged pictures from the first super-resolution enlarger as a set of encoding target pictures, employing the set of super-resolution enlarged decoded pictures from the second super-resolution enlarger and the set of resolution converted enlarged decoded pictures from the third resolution converter as sets of reference pictures, to implement thereon a third combination of processes for a prediction and an encoding to create a third sequence of encoded bits. For example, when an I-frame is encoded, the encoder acts as a third encoder, using two reference picture sets (i.e., p-reference-frames and downscaled/upscaled reconstruction reference frames). The p-reference-frames (i.e., first reference picture set that is the output of the first encoder) denotes the list of reference pictures consisting of input pictures at the original resolution. On the other hand, reconstruction reference frames (i.e., second reference set that is the output of second resolution converter) denotes the reference picture list which contains the reconstructed pictures.

```
// Encode frames.
while (aom_img_read(raw, infile) && frame_count < limit) {
  ++frame_count;
  encode_frame(&codec, raw, frame_count, 1, 0, writer);
}
```

Source: https://github.com/ultrawide/libaom/blob/master/examples/twopass_encoder.c

```
void av1_encode_mv(AV1_COMP *cpi, aom_writer *w, const MV *mv, const MV *ref,
                   nmv_context *mvctx, int usehp);
```

Source: https://github.com/ultrawide/libaom/blob/master/av1/encoder/encodemv.h

```
static int encode_frame(aom_codec_ctx_t *ctx, const aom_image_t *img,
                        aom_codec_pts_t pts, unsigned int duration,
                        aom_enc_frame_flags_t flags, AvxVideoWriter *writer) {
  int got_pkts = 0;
  aom_codec_iter_t iter = NULL;
  const aom_codec_cx_pkt_t *pkt = NULL;
  const aom_codec_err_t res = aom_codec_encode(ctx, img, pts, duration, flags);
  if (res != AOM_CODEC_OK) die_codec(ctx, "Failed to encode frame.");

  while ((pkt = aom_codec_get_cx_data(ctx, &iter)) != NULL) {
    got_pkts = 1;
    if (pkt->kind == AOM_CODEC_CX_FRAME_PKT) {
      const int keyframe = (pkt->data.frame.flags & AOM_FRAME_IS_KEY) != 0;

      if (!aom_video_writer_write_frame(writer, pkt->data.frame.buf,
                                        pkt->data.frame.sz,
                                        pkt->data.frame.pts))
        die_codec(ctx, "Failed to write compressed frame.");
      printf(keyframe ? "K" : ".");
      fflush(stdout);
    }
```

Source: https://github.com/ultrawide/libaom/blob/master/examples/twopass_encoder.c

```
  // Encode frames.
  while (aom_img_read(&raw, infile)) {
    int flags = 0;
    if (keyframe_interval > 0 && frame_count % keyframe_interval == 0)
      flags |= AOM_EFLAG_FORCE_KF;
    encode_frame(&codec, &raw, frame_count++, flags, writer);
    frames_encoded++;
    if (max_frames > 0 && frames_encoded >= max_frames) break;
  }

  // Flush encoder.
  while (encode_frame(&codec, NULL, -1, 0, writer)) continue;

  printf("\n");
  fclose(infile);
  printf("Processed %d frames.\n", frame_count);

  aom_img_free(&raw);
  if (aom_codec_destroy(&codec)) die_codec(&codec, "Failed to destroy codec.");

  aom_video_writer_close(writer);

  return EXIT_SUCCESS;
}
```

Source: https://github.com/ultrawide/libaom/blob/master/examples/simple_encoder.c

154. The ’448 Accused Products include wherein a spatial resolution of the set of super-resolution enlarged pictures, that of the set of super-resolution enlarged decoded pictures, and that of the set of resolution converted enlarged decoded pictures are made equal. As discussed above, the AV1 encoding process uses downscaled input images which are upscaled using upscaled reconstructed reference images (i.e., reference pictures) and a linear upscale process after deblocking and CDEF steps. Thus, the encoder converts the resolution of applied downscaled input images at various stages to generate the respective upscaled output images (i.e., super-resolution enlarged pictures, super-resolution enlarged decoded pictures, resolution converted enlarged decoded pictures). Since the upscaling is done on the output of all three stages, the resolution of the output images generated via all of the three processes is equal.

3.7.4 Frame super-resolution

To improve the perceptual quality of decoded pictures, a super-resolution process can be applied at low bitrates. The super-resolution process can be performed on a per-frame basis. First, at the encoder side, the source video is downscaled as a nonnormative procedure. Second, the downscaled video is encoded, followed by deblocking and CDEF filtering processes. Third, a linear upscaling process is applied as a normative procedure to convert the encoded video back to its original resolution. Finally, an LR filter is applied to further recover some lost details of the upscaled picture. The last two steps make up the super-resolving process; the deblocking and CDEF filters are applied at lower spatial resolution on the decoder side, after which the frames are passed through the super-resolution process. To reduce complexity regarding the use of line buffers in hardware implementation, the upscaling and downscaling processes are applied to the horizontal dimension only.

Source**:** https://aomedia.org/docs/AV1_ToolDescription_v11-clean.pdf, 36

155. The ’448 Accused Products include a third encoder which controls a selection of a set of reference pictures and creates a set of data on the selection of the set of reference pictures to identify a selected set of reference pictures during the process for prediction of the third combination. As discussed above, the AV1 encoding process uses two sets of reference frames. Since different processes (e.g., motion estimation and mode decision) in encoding use reference pictures from “pa_ref” (i.e., set of first reference pictures) and “recon_ref” (i.e., set of second reference pictures) respectively, it appears that the encoder selects one reference frame from each of the first and second reference picture set, which further constitutes the set of selected reference frames. Further, the reference frame from each first and second set is selected based on the denominator value, which matches the width and current resolution of the input picture.

```
static int choose_primary_ref_frame(
    const AV1_COMP *const cpi, const EncodeFrameParams *const frame_params) {
  const AV1_COMMON *const cm = &cpi->common;

  const int intra_only = frame_params->frame_type == KEY_FRAME ||
                         frame_params->frame_type == INTRA_ONLY_FRAME;
  if (intra_only || frame_params->error_resilient_mode || cpi->use_svc ||
      cpi->ext_flags.use_primary_ref_none) {
    return PRIMARY_REF_NONE;
  }
```

Source: https://github.com/ultrawide/libaom/blob/master/av1/encoder/encode_strategy.c

```
static int get_current_frame_ref_type(
    const AV1_COMP *const cpi, const EncodeFrameParams *const frame_params) {
  // We choose the reference "type" of this frame from the flags which indicate
  // which reference frames will be refreshed by it.  More than one  of these
  // flags may be set, so the order here implies an order of precedence. This is
  // just used to choose the primary_ref_frame (as the most recent reference
  // buffer of the same reference-type as the current frame)
```

Source: https://github.com/ultrawide/libaom/blob/master/av1/encoder/encode_strategy.c

```
void av1_find_best_ref_mvs_from_stack(int allow_hp,
                                      const MB_MODE_INFO_EXT *mbmi_ext,
                                      MV_REFERENCE_FRAME ref_frame,
                                      int_mv *nearest_mv, int_mv *near_mv,
                                      int is_integer);
```

Source: https://github.com/ultrawide/libaom/blob/master/av1/encoder/encode_strategy.c

156. Google has indirectly infringed and continues to indirectly infringe one or more claims of the '448 Patent, as provided by 35 U.S.C. § 271(b), by inducing infringement by others, such as Google's customers and end-users, in this Judicial District and elsewhere in the United States. For example, Google's customers and end-users directly infringe, either literally or under the doctrine of equivalents, through their use of the inventions claimed in the '448 Patent. Google induces this direct infringement through its affirmative acts of manufacturing, selling, distributing, and/or otherwise making available the '448 Accused Products, and providing instructions, documentation, and other information to customers and end-users suggesting that they use the '448 Accused Products in an infringing manner, including technical support, marketing, product manuals, advertisements, and online documentation.[45] Because of Google's inducement, Google's customers and end-users use the '448 Accused Products in a way Google intends and they directly

[45] *See, e.g.*, https://guidebooks.google.com/home; https://pixel.withgoogle.com/?hl=en&country=US; https://pixel.withgoogle.com/support/Pixel?hl=en&country=US&dark=0; https://www.youtube.com/@Googlehelp/playlists

infringe the ’448 Patent. Google performs these affirmative acts with knowledge of the ’448 Patent and with the intent, or willful blindness, that the induced acts directly infringe the ’448 Patent.

157. Google has indirectly infringed and continues to indirectly infringe one or more claims of the ’448 Patent, as provided by 35 U.S.C. § 271(c), by contributing to direct infringement by others, such as customers and end-users, in this Judicial District and elsewhere in the United States. Google’s affirmative acts of selling and offering to sell the ’448 Accused Products in this Judicial District and elsewhere in the United States and causing the ’448 Accused Products to be manufactured, used, sold, and offered for sale contribute to others’ use and manufacture of the ’448 Accused Products, such that the ’448 Patent is directly infringed by others.[46] The accused components within the ’448 Accused Products including, but not limited to, software manufactured by Google, are material to the invention of the ’448 Patent, are not staple articles or commodities of commerce, have no substantial non-infringing uses, and are known by Google to be especially made or adapted for use in the infringement of the ’448 Patent. Google performs these affirmative acts with knowledge of the ’448 Patent and with intent, or willful blindness, that they cause the direct infringement of the ’448 Patent.

158. Google’s infringement of the ’448 Patent is and has been willful. Google was on notice of the existence of the ’448 Patent and its infringement thereof, or has been willfully blind as to the existence of the ’448 Patent and its infringement thereof. As one example, Google is a founding member of the Alliance for Open Media, the organization that publishes the AV1 Specification, as of the time of its inception in 2015. Google’s Head of Strategy and Partnerships for Chrome Media, Matt Frost, is currently listed as the Chair of AOM’s Steering Committee. The Alliance for Open Media’s stated goal was to create a video codec that was free of patent licensing

[46] *See, e.g.*, https://store.google.com/magazine/support?hl=en-US

obligations associated with prior video codecs.[47] For example, Google's YouTube product previously used a video codec called HEVC, and Google was motivated to avoid HEVC licensing fees by developing AV1 through the Alliance for Open Media.[48] The Alliance for Open Media, including Google, conducted a "comprehensive evaluation of the video codec patent landscape and performance of patent due diligence by world-class codec engineers and legal professionals during the development stage."[49] Upon information and belief, this "patent due diligence" either uncovered the existence of the '448 Patent and Google's infringement thereof, or should have uncovered the existence of the '448 Patent and Google's infringement thereof. Google could not have reasonably believed that the development of the AV1 video codec could not infringe any valid patent claims, including those of the '448 Patent.

159. Defendant has been on actual notice of the '448 Patent and Defendant's infringement thereof at least as of June 3, 2013, when it was cited during prosecution of U.S. Patent No. 8,635,357, entitled "Dynamic selection of parameter sets for transcoding media data," and currently assigned to Google LLC. Specifically, during prosecution, the Examiner cited to U.S. Patent Publication No. 2011/0075734 A1, which is the published patent application number of the '448 Patent.

160. ACT has suffered damages as a result of Defendant's direct, indirect, and willful infringement of the '448 Patent in an amount to be proved at trial.

---

[47] https://aomedia.org/press%20releases/alliance-to-deliver-next-generation-open-media-formats/
[48] https://arstechnica.com/information-technology/2015/09/microsoft-google-amazon-others-aim-for-royalty-free-video-codecs/
[49] https://aomedia.org/press%20releases/the-alliance-for-open-media-statement/

COUNT V
(Infringement of the ’101 Patent)

161. Paragraphs 1 through 90 are incorporated by reference as if fully set forth herein.

162. ACT has not licensed or otherwise authorized Defendant to make, use, offer for sale, sell, or import any products that embody the inventions of the ’101 Patent.

163. Defendant has and continues to directly infringe the ’101 Patent, either literally or under the doctrine of equivalents, without authority and in violation of 35 U.S.C. § 271, by making, using, offering to sell, selling, and/or importing into the United States products that satisfy each and every limitation of one or more claims of the ’101 Patent. Such products include at least Google’s systems that perform network content delivery, including Youtube, which utilize one or more video codecs for network content delivery, Defendant’s Google Cloud Content Delivery Network (CDN), and Defendant’s Google Smart Home (the ’101 Accused Products), which includes a server device for media, the server device for media comprising: an internal storage device for storing digital contents, wherein the server device for media responds to a data transmission request from a network player by stream-delivering corresponding data in corresponding digital contents from the internal storage device to the network player during connection to a network; a transfer control unit adapted to transfer and store part of held digital contents in the internal storage device to a network storage device, wherein the network storage device is connected to the network and is capable of storing data, and wherein said transfer control unit does not transfer, from the internal storage device to the network storage device, the digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device; a list information transmission unit adapted to respond to a list presentation request for the held digital contents of the server device for media from the network player by transmitting list information to the network

player, wherein the list information lists the digital contents left in the internal storage device and the digital contents transferred from the internal storage device to the network storage device and stored in the network storage device, and wherein the list information maintains a tree structure of the digital contents in the internal storage device before transferring the digital contents to the network storage device; a search unit adapted to respond to a data transmission request for the held digital contents from the network player by searching for a location where the held digital contents are currently stored; and a digital contents data transmission processing unit adapted to allow the corresponding data in held digital contents to be stream-delivered from the network storage device to the network player, if the result of search shows the network storage device, wherein the server device for media is a media player.

164. For example, Defendant has and continues to directly infringe at least claim 1 of the ’101 Patent by making, using, offering to sell, selling, and/or importing into the United States the ’101 Accused Products.

165. The ’101 Accused Products include a server device for media, the server device for media comprising an internal storage device for storing digital contents. For example, Google Cloud CDN includes a server device for the streaming/delivery of digital content (i.e., A server device for media) such as videos, images, etc., stored at Google’s Cloud Server (i.e., server device for media) to user devices via APIs (i.e., Network Player). The digital content that is delivered is stored at Google’s cloud storage servers deployed across the globe. These cloud servers include cloud storage (i.e., an internal storage device) to save digital content such as videos, images, etc.

What is Cloud CDN?

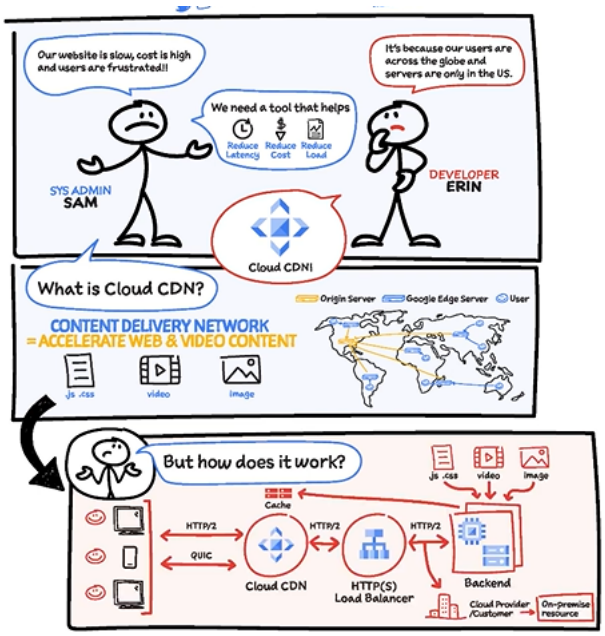
Cloud CDN is a content delivery network that accelerates your web and video content delivery by using Google's global edge network to bring content as close to your users as possible. As a result latency, cost, and load on your backend servers is reduced, making it easier to scale to millions of users. Global anycast IP provides a single IP for global reach. It enables Google Cloud to route users to the nearest edge cache automatically and avoid DNS propagation delays that can impact availability. It supports HTTP/2 end-to-end and the QUIC protocol from client to cache. QUIC is a multiplexed stream transport over UDP, which reduces latency and makes it ideal for lossy mobile networks.

Source: https://cloud.google.com/learn/what-is-cloud-storage?hl=en

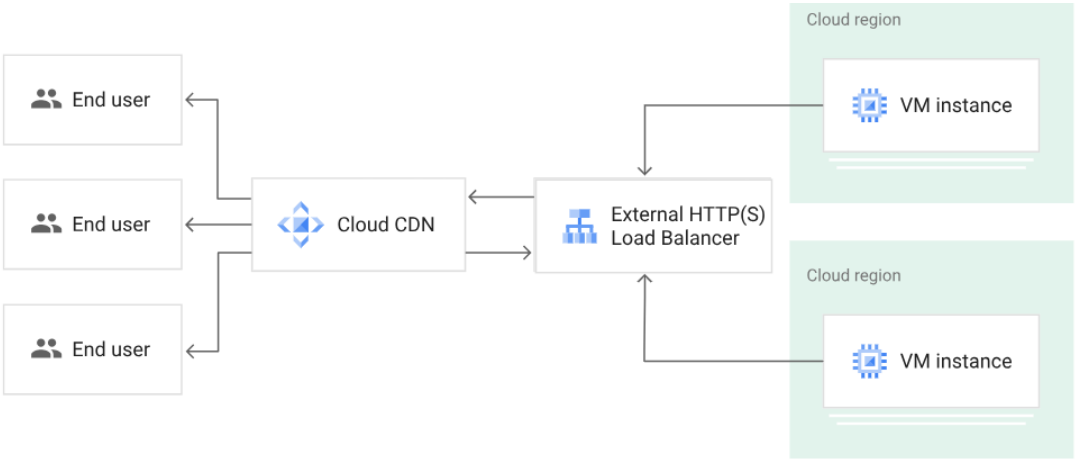
Content delivery

With the ability to save copies of media data, such as large audio and video files, on servers dispersed across the globe, media and entertainment companies can serve their audience low-latency, always available content from wherever they reside.

Source: https://cloud.google.com/learn/what-is-cloud-storage?hl=en



Source: https://cloud.google.com/blog/topics/developers-practitioners/what-cloud-cdn-and-how-does-it-work



Source: https://cloud.google.com/cdn/docs/overview

How does Cloud Storage work?

Cloud Storage uses remote servers to save data, such as files, business data, videos, or images. Users upload data to servers via an internet connection, where it is saved on a virtual machine on a physical server. To maintain availability and provide redundancy, cloud providers will often spread data to multiple virtual machines in data centers located across the world. If storage needs increase, the cloud provider will spin up more virtual machines to handle the load. Users can access data in Cloud Storage through an internet connection and software such as web portal, browser, or mobile app via an application programming interface (API).

Cloud Storage is available in four different models:

Source: https://cloud.google.com/learn/what-is-cloud-storage?hl=en

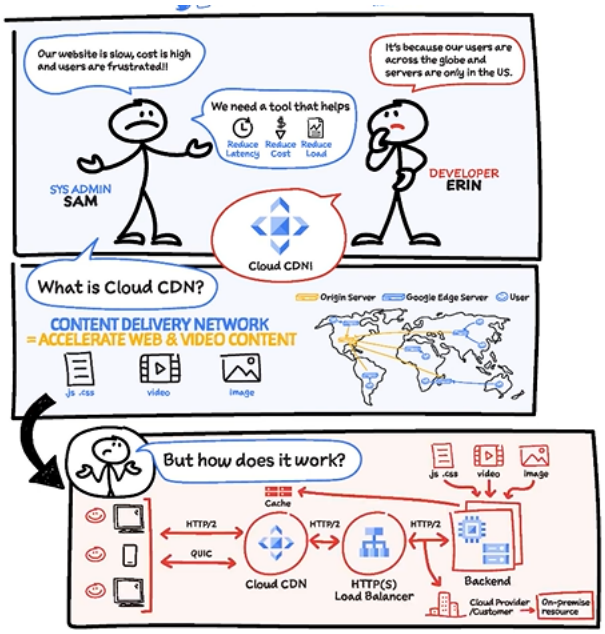
166. The ’101 Accused Products include a server device for media responding to a data transmission request from a network player by stream-delivering corresponding data in corresponding digital contents from the internal storage device to the network player during connection to a network. For example, when Google Cloud CDN receives a request for digital content via APIs and browsers, the request is handled by frontend APIs. In response to the request for accessing the digital content from the user device, the Google Cloud Storage servers (i.e., server devices for media) deliver the contents by first transferring the requested data from server storage to the CDN cache (i.e., network device storage). The cache storage of the CDN network streams this data via various APIs and browsers (i.e., Network Player).

How Cloud CDN works

When a user requests content from an external Application Load Balancer, the request arrives at a GFE that is at the edge of Google's network as close as possible to the user.

If the load balancer's URL map routes traffic to a backend service or backend bucket that has Cloud CDN configured, the GFE uses Cloud CDN.

Source: https://cloud.google.com/cdn/docs/overview



Source: https://cloud.google.com/blog/topics/developers-practitioners/what-cloud-cdn-and-how-does-it-work

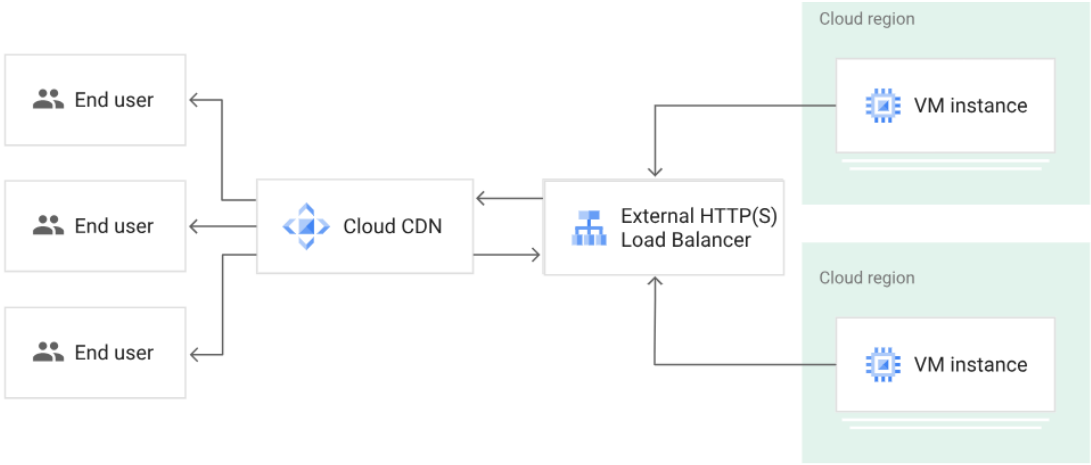
Cloud CDN works with the global external Application Load Balancer or the classic Application Load Balancer to deliver content to your users. The external Application Load Balancer provides the frontend IP addresses and ports that receive requests and the backends that respond to the requests.

Source: https://cloud.google.com/cdn/docs/overview

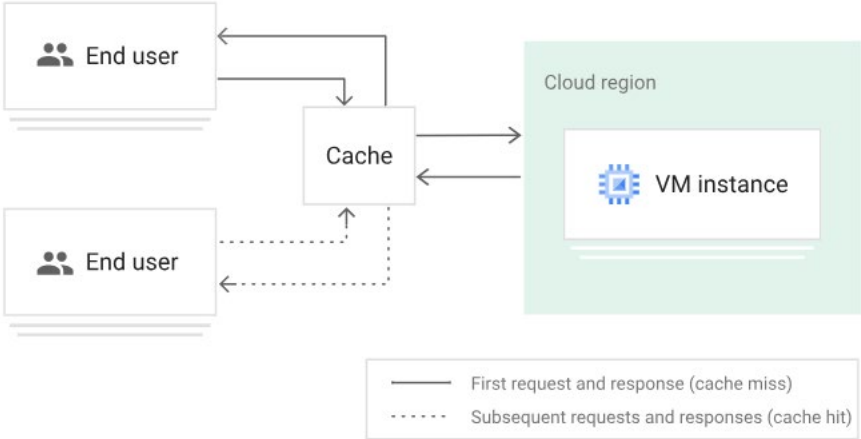
Cloud CDN content can be sourced from various types of backends.

In Cloud CDN, these backends are also called *origin servers*. Figure 1 illustrates how responses from origin servers that run on virtual machine (VM) instances flow through an external Application Load Balancer before being delivered by Cloud CDN. In this situation, the Google Front End (GFE) comprises Cloud CDN and the external Application Load Balancer.

Source: https://cloud.google.com/cdn/docs/overview



Source: https://cloud.google.com/cdn/docs/overview



Source: https://cloud.google.com/cdn/docs/overview

167. The '101 Accused Products include a transfer control unit adapted to transfer and store part of held digital contents in the internal storage device to a network storage device, wherein the network storage device is connected to the network and is capable of storing data, and wherein said transfer control unit does not transfer, from the internal storage device to the network storage device, the digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device. For example, when Google Cloud CDN receives a request for data, the Google Cloud storage servers first cache (i.e., transfer and store) the data from cloud storage (i.e., internal storage) to

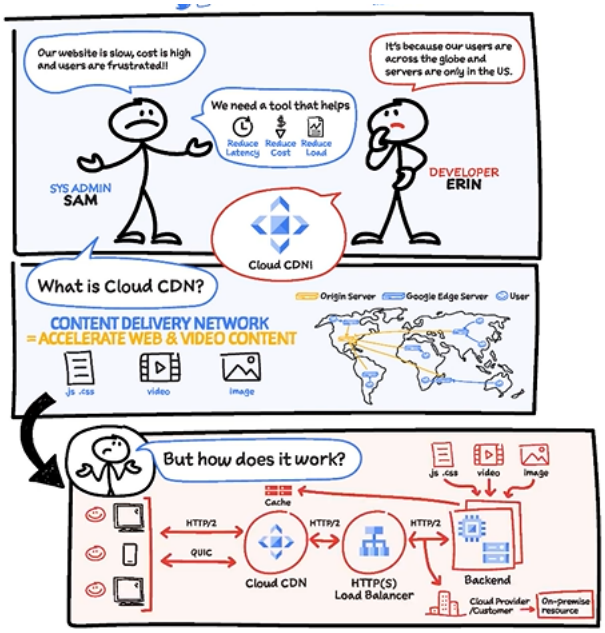
Cloud CDN storage (i.e., network storage). While delivering the data from cloud server to various APIs, the data is only cached to the cloud CDN storage if it is cacheable (i.e., wherein the digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents are not transferred from the internal storage device to the network storage device")

How Cloud CDN works

When a user requests content from an external Application Load Balancer, the request arrives at a GFE that is at the edge of Google's network as close as possible to the user.

If the load balancer's URL map routes traffic to a backend service or backend bucket that has Cloud CDN configured, the GFE uses Cloud CDN.

Source: https://cloud.google.com/cdn/docs/overview



Source: https://cloud.google.com/blog/topics/developers-practitioners/what-cloud-cdn-and-how-does-it-work

Cloud CDN overview

Send feedback

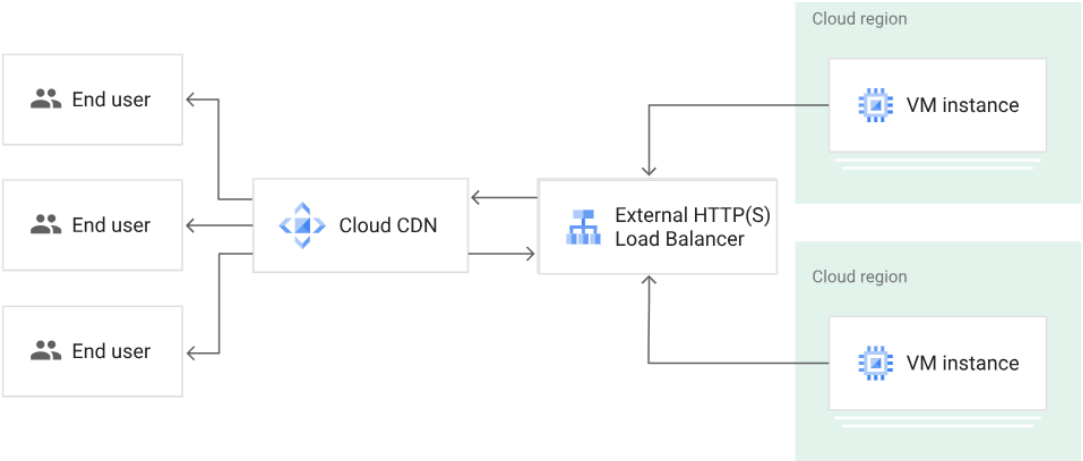
Cloud CDN (Content Delivery Network) uses Google's global edge network to serve content closer to users, which accelerates your websites and applications.

Cloud CDN works with the global external Application Load Balancer or the classic Application Load Balancer to deliver content to your users. The external Application Load Balancer provides the frontend IP addresses and ports that receive requests and the backends that respond to the requests.

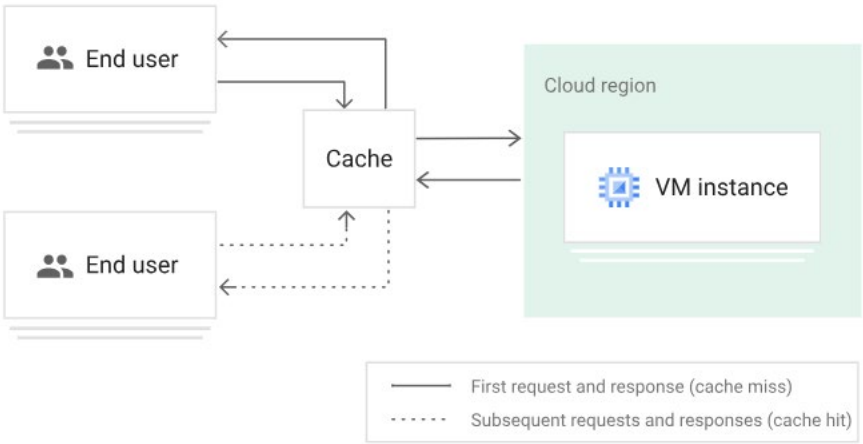
Cloud CDN content can be sourced from various types of backends.

In Cloud CDN, these backends are also called *origin servers*. Figure 1 illustrates how responses from origin servers that run on virtual machine (VM) instances flow through an external Application Load Balancer before being delivered by Cloud CDN. In this situation, the Google Front End (GFE) comprises Cloud CDN and the external Application Load Balancer.

Source: https://cloud.google.com/cdn/docs/overview



Source: https://cloud.google.com/cdn/docs/overview



Source: https://cloud.google.com/cdn/docs/overview

How Cloud CDN works

When a user requests content from an external Application Load Balancer, the request arrives at a GFE that is at the edge of Google's network as close as possible to the user.

If the load balancer's URL map routes traffic to a backend service or backend bucket that has Cloud CDN configured, the GFE uses Cloud CDN.

Cache hits and cache misses

A cache is a group of servers that stores and manages content so that future requests for that content can be served faster. The cached content is a copy of cacheable content that is stored on origin servers.

If the GFE looks in the Cloud CDN cache and finds a cached response to the user's request, the GFE sends the cached response to the user. This is called a *cache hit*. When a cache hit occurs, the GFE looks up the content by its cache key and responds directly to the user, shortening the round-trip time and saving the origin server from having to process the request.

A *partial hit* occurs when a request is served partially from cache and partially from a backend. This can happen if only part of the requested content is stored in a Cloud CDN cache, as described in Support for byte range requests.

Source: https://cloud.google.com/cdn/docs/overview

The first time that a piece of content is requested, the GFE determines that it can't fulfill the request from the cache. This is called a *cache miss*. When a cache miss occurs, the GFE forwards the request to the external Application Load Balancer. The load balancer then forwards the request to one of your origin servers. When the cache receives the content, the GFE forwards the content to the user.

If the origin server's response to this request is cacheable, Cloud CDN stores the response in the Cloud CDN cache for future requests. Data transfer from a cache to a client is called *cache egress*. Data transfer to a cache is called *cache fill*.

Figure 2 shows a cache hit and a cache miss:

1. Origin servers running on VM instances send HTTP(S) responses.
2. The external Application Load Balancer distributes the responses to Cloud CDN.
3. Cloud CDN delivers the responses to end users.

Source: https://cloud.google.com/cdn/docs/overview

Cache hit ratio

The *cache hit ratio* is the percentage of times that a requested object is served from the cache. If the cache hit ratio is 60%, it means that the requested object is served from the cache 60% of the time and must be retrieved from the origin 40% of the time.

For information about how cache keys can affect the cache hit ratio, see Using cache keys. For troubleshooting information, see Cache hit ratio is low.

Source: https://cloud.google.com/cdn/docs/overview

Inserting content into the cache

Caching is reactive in that an object is stored in a particular cache if a request goes through that cache and if the response is cacheable. An object stored in one cache does not automatically replicate into other caches; cache fill happens only in response to a client-initiated request. You cannot preload caches except by causing the individual caches to respond to requests.

When the origin server supports byte range requests, Cloud CDN can initiate multiple cache fill requests in reaction to a single client request.

Source: https://cloud.google.com/cdn/docs/overview

Serving content from a cache

After you enable Cloud CDN, caching happens automatically for all cacheable content. Your origin server uses HTTP headers to indicate which responses are cached. You can also control cacheability by using cache modes.

When you use a backend bucket, the origin server is Cloud Storage. When you use VM instances, the origin server is the web server software that you run on those instances.

Cloud CDN uses caches in numerous locations around the world. Because of the nature of caches, it is impossible to predict whether a particular request is served out of a cache. You can, however, expect that popular requests for cacheable content are served from a cache most of the time, yielding significantly reduced latencies, reduced cost, and reduced load on your origin servers.

For more information about what Cloud CDN caches and for how long, see the Caching overview.

To see what Cloud CDN is serving from a cache, you can view logs.

Source: https://cloud.google.com/cdn/docs/overview

168. The ’101 Accused Products include a list information transmission unit adapted to respond to a list presentation request for the held digital contents of the server device for media from the network player by transmitting list information to the network player, wherein the list information lists the digital contents left in the internal storage device and the digital contents transferred from the internal storage device to the network storage device and stored in the network storage device, and wherein the list information maintains a tree structure of the digital contents in the internal storage device before transferring the digital contents to the network storage device. For example, when Google Cloud CDN receives a request to access a playlist of the content via APIs (e.g., requesting a YouTube playlist), in responding to that request, the user API (i.e., Network Player such as You Tube) is presented with a list of digital content. The list shows (i.e., list information lists) the digital content stored at the Google Cloud storage (i.e., content left in an internal storage device) and the digital content stored at the Google Cloud CDN (i.e., Network Storage). Since the information packets are retrieved and structured in the form of a user playlist, the present information is in the form of a tree structure. For example, the packet capture images

below depict that when a user requested a digital content via the user interface, the response presented a list of packets that came from two distinct locations. Packet number 622, which is directed from a url _cloud-netblocks .googleusercontent.com, according to a conversation by the Google support platform this URL, represents the content delivery network, whereas packet 317 is received from the YouTube backend server. Therefore the the list presents the content from both the server storage device as well as the network storage device.

Stevo Novkovski Jun 15, 2016, 6:44:02 AM
to gce-discussion

Hello,

How we can white-list google CDN in our backend?
There is no ip-range list.

George (Google Cloud Support) Jun 15, 2016, 8:12:13 PM
to gce-discussion

Hello Stevo,

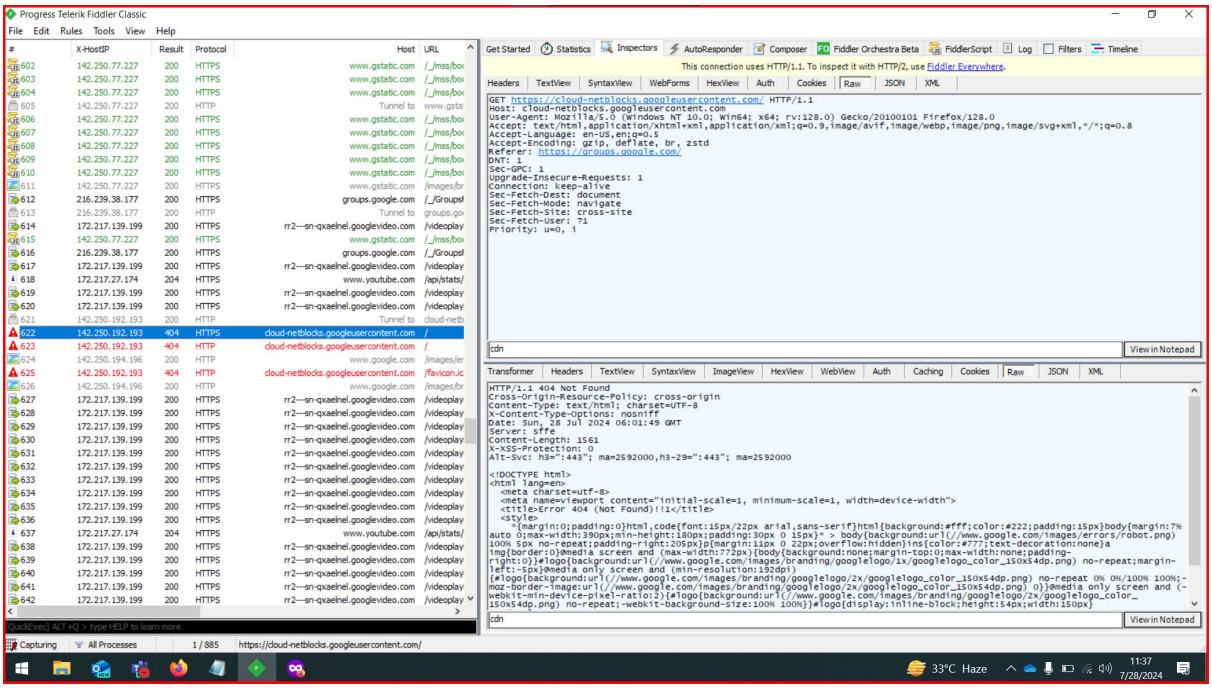
Google Cloud Platform uses a large range of IP addresses, which change over time. For historical reasons, Google Cloud Platform publishes its list of public IP addresses in an SPF record for `_cloud-netblocks.googleusercontent.com`.

When you need the literal IP addresses for Google Cloud Platform, use one of the common DNS lookup commands (`nslookup`, `dig`, `host`) to retrieve the TXT records for the domain `_cloud-netblocks.googleusercontent.com`:

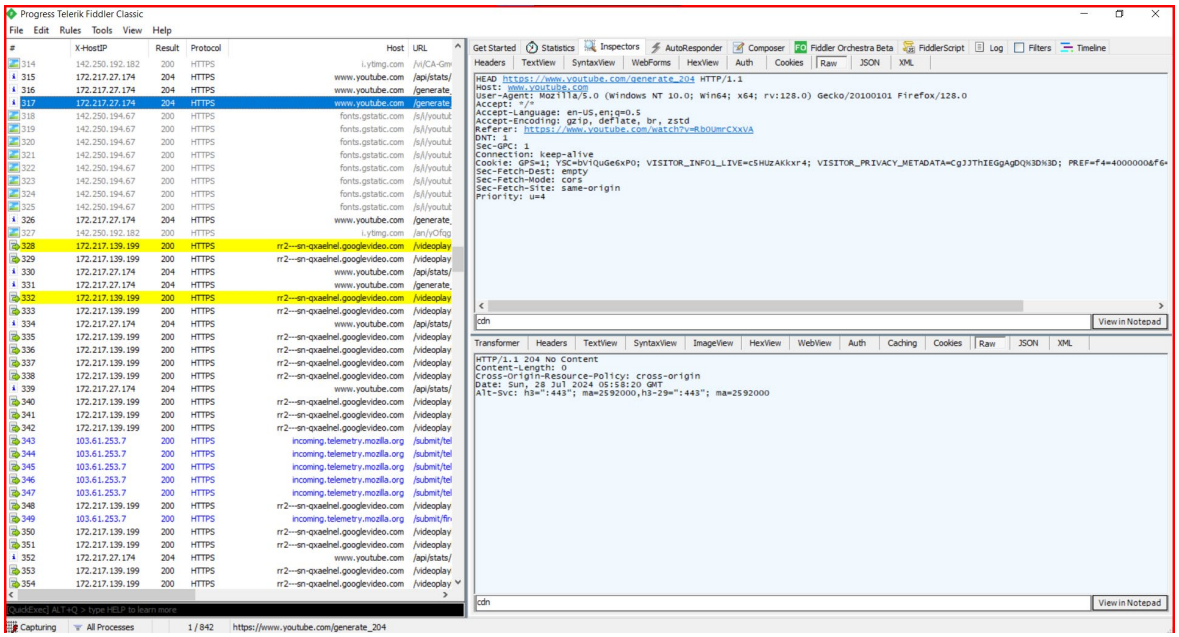
```
$ nslookup -q=TXT _cloud-netblocks.googleusercontent.com  8.8.8.8
```

This returns a list of the domains included in Google's SPF record, such as:

Source: https://cloud.google.com/cdn/docs/overview



Source: You Tube Packet Capture for Accessing a user playlist



Source: You Tube Packet Capture for Accessing a user playlist

169. The '101 Accused Products include a search unit adapted to respond to a data transmission request for the held digital contents from the network player by searching for a location where the held digital contents are currently stored. For example, when Google Cloud CDN receives a request to access

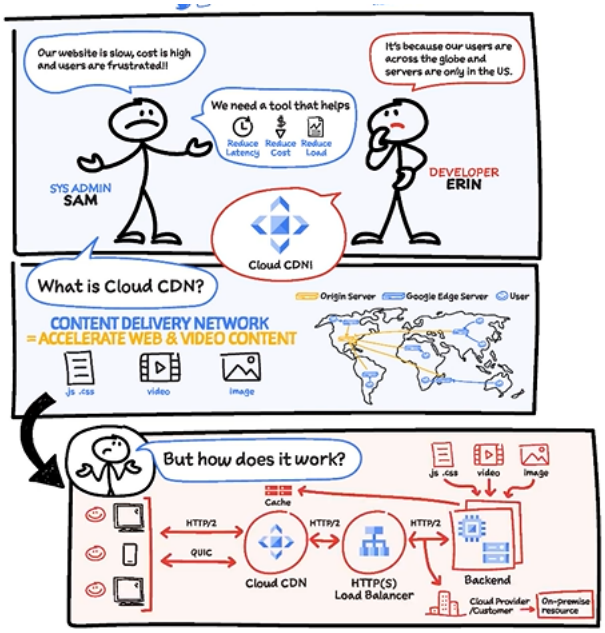
the digital content via APIs, in responding to that request from the user API (i.e., Network Player), the content is first searched across the CDN cache. If the search in cache results in a “cache miss,” the content is searched at the original Google server storage. For example, Packet 622, highlighted below, is received in response to the content access request, whereas packet 317, highlighted in the figure below that one, is received from the Google Cloud CDN. This indicates that when a request to access content via APIs is made, the content is searched across both the backend servers and the CDN, and then retrieved from its location.

How Cloud CDN works

When a user requests content from an external Application Load Balancer, the request arrives at a GFE that is at the edge of Google's network as close as possible to the user.

If the load balancer's URL map routes traffic to a backend service or backend bucket that has Cloud CDN configured, the GFE uses Cloud CDN.

Source: https://cloud.google.com/cdn/docs/overview



Source: https://www.youtube.com/watch?v=EumuFAfTWJY

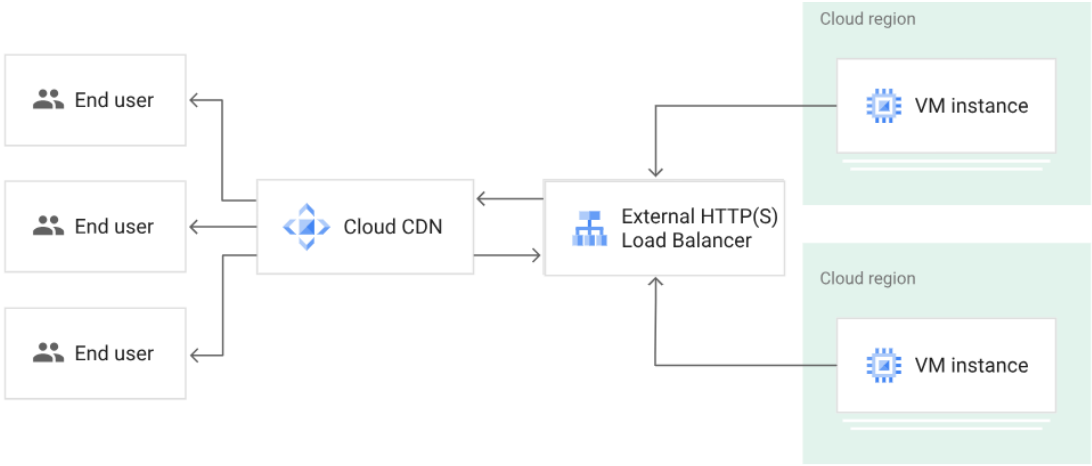
Cloud CDN works with the global external Application Load Balancer or the classic Application Load Balancer to deliver content to your users. The external Application Load Balancer provides the frontend IP addresses and ports that receive requests and the backends that respond to the requests.

Source: https://cloud.google.com/cdn/docs/overview

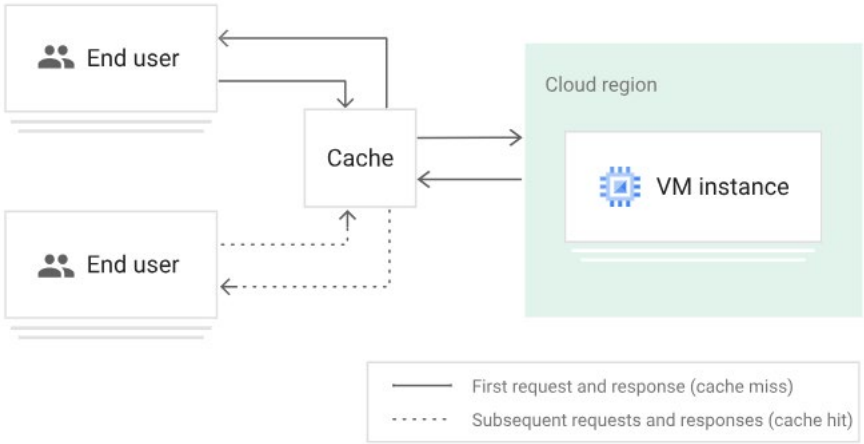
Cloud CDN content can be sourced from various types of backends.

In Cloud CDN, these backends are also called *origin servers*. Figure 1 illustrates how responses from origin servers that run on virtual machine (VM) instances flow through an external Application Load Balancer before being delivered by Cloud CDN. In this situation, the Google Front End (GFE) comprises Cloud CDN and the external Application Load Balancer.

Source: https://cloud.google.com/cdn/docs/overview



Source: https://cloud.google.com/cdn/docs/overview



Source: https://cloud.google.com/cdn/docs/overview

The first time that a piece of content is requested, the GFE determines that it can't fulfill the request from the cache. This is called a *cache miss*. When a cache miss occurs, the GFE forwards the request to the external Application Load Balancer. The load balancer then forwards the request to one of your origin servers. When the cache receives the content, the GFE forwards the content to the user.

If the origin server's response to this request is cacheable, Cloud CDN stores the response in the Cloud CDN cache for future requests. Data transfer from a cache to a client is called *cache egress*. Data transfer to a cache is called *cache fill*.

Figure 2 shows a cache hit and a cache miss:

1. Origin servers running on VM instances send HTTP(S) responses.
2. The external Application Load Balancer distributes the responses to Cloud CDN.
3. Cloud CDN delivers the responses to end users.

Source: https://cloud.google.com/cdn/docs/overview

Cache hit ratio

The *cache hit ratio* is the percentage of times that a requested object is served from the cache. If the cache hit ratio is 60%, it means that the requested object is served from the cache 60% of the time and must be retrieved from the origin 40% of the time.

For information about how cache keys can affect the cache hit ratio, see Using cache keys. For troubleshooting information, see Cache hit ratio is low.

Source: https://cloud.google.com/cdn/docs/overview

Serving content from a cache

After you enable Cloud CDN, caching happens automatically for all cacheable content. Your origin server uses HTTP headers to indicate which responses are cached. You can also control cacheability by using cache modes.

When you use a backend bucket, the origin server is Cloud Storage. When you use VM instances, the origin server is the web server software that you run on those instances.

Cloud CDN uses caches in numerous locations around the world. Because of the nature of caches, it is impossible to predict whether a particular request is served out of a cache. You can, however, expect that popular requests for cacheable content are served from a cache most of the time, yielding significantly reduced latencies, reduced cost, and reduced load on your origin servers.

For more information about what Cloud CDN caches and for how long, see the Caching overview.

To see what Cloud CDN is serving from a cache, you can view logs.

Source: https://cloud.google.com/cdn/docs/overview

Stevo Novkovski Jun 15, 2016, 6:44:02 AM

to gce-discussion

Hello,

How we can white-list google CDN in our backend?
There is no ip-range list.

George (Google Cloud Support) Jun 15, 2016, 8:12:13 PM

to gce-discussion

Hello Stevo,

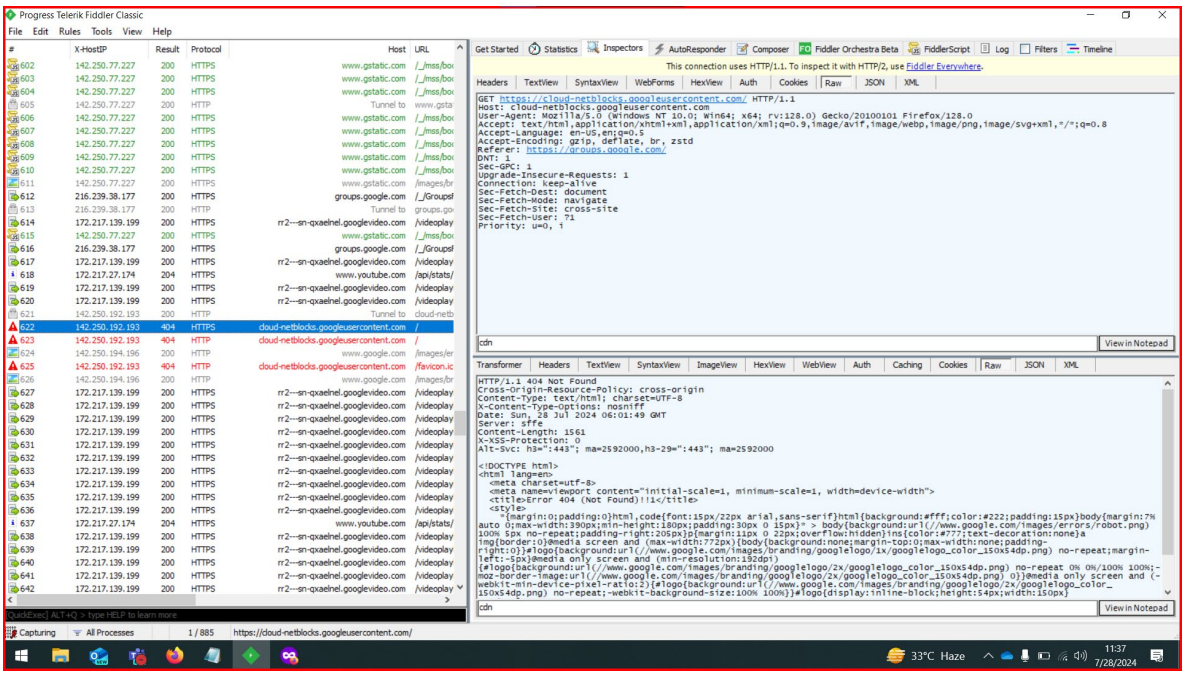
Google Cloud Platform uses a large range of IP addresses, which change over time. For historical reasons, Google Cloud Platform publishes its list of public IP addresses in an SPF record for `_cloud-netblocks.googleusercontent.com`.

When you need the literal IP addresses for Google Cloud Platform, use one of the common DNS lookup commands (`nslookup`, `dig`, `host`) to retrieve the TXT records for the domain `_cloud-netblocks.googleusercontent.com`:

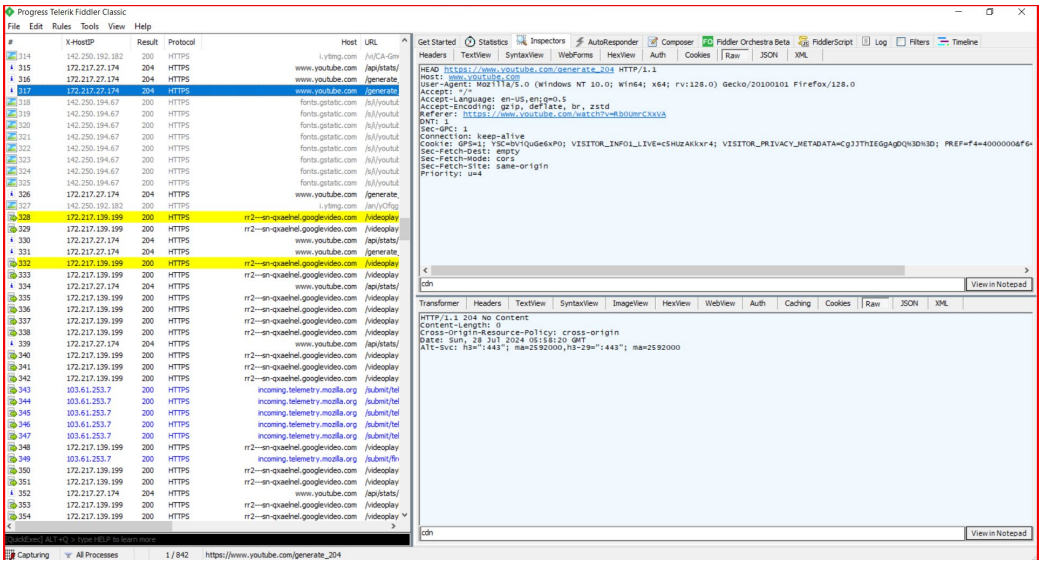
```
$ nslookup -q=TXT _cloud-netblocks.googleusercontent.com  8.8.8.8
```

This returns a list of the domains included in Google's SPF record, such as:

Source: https://groups.google.com/g/gce-discussion/c/V2n9Ri-T5qg



Source: You Tube Packet Capture for Accessing a user playlist



Source: You Tube Packet Capture for Accessing a User Playlist

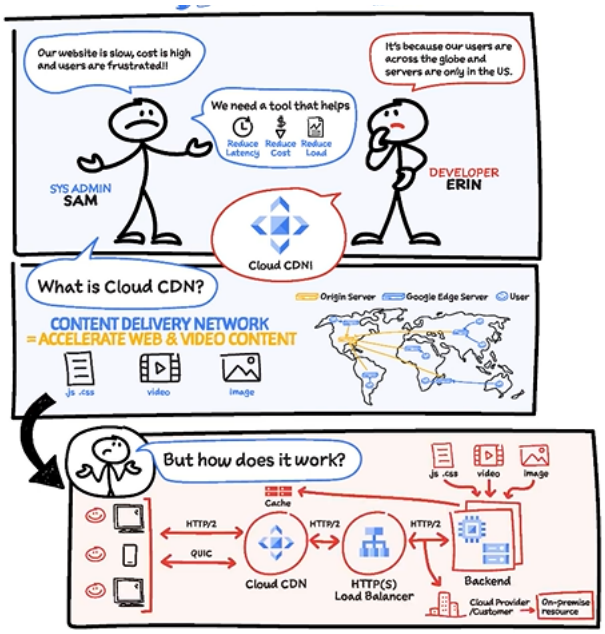
170. The ’101 Accused Products include a digital contents data transmission processing unit adapted to allow the corresponding data in held digital contents to be stream-delivered from the network storage device to the network player, if the result of search shows the network storage device. For example, when digital content is requested from Google Cloud CDN via APIs (i.e., Network Player) that digital content is present at Google Cloud CDN, that results in a “cache hit.” In this case, the user API (i.e., Network Player) accesses and delivers the digital content directly from Google Cloud CDN (i.e., Network Storage).

How Cloud CDN works

When a user requests content from an external Application Load Balancer, the request arrives at a GFE that is at the edge of Google's network as close as possible to the user.

If the load balancer's URL map routes traffic to a backend service or backend bucket that has Cloud CDN configured, the GFE uses Cloud CDN.

Source: https://cloud.google.com/cdn/docs/overview



Source: https://cloud.google.com/blog/topics/developers-practitioners/what-cloud-cdn-and-how-does-it-work

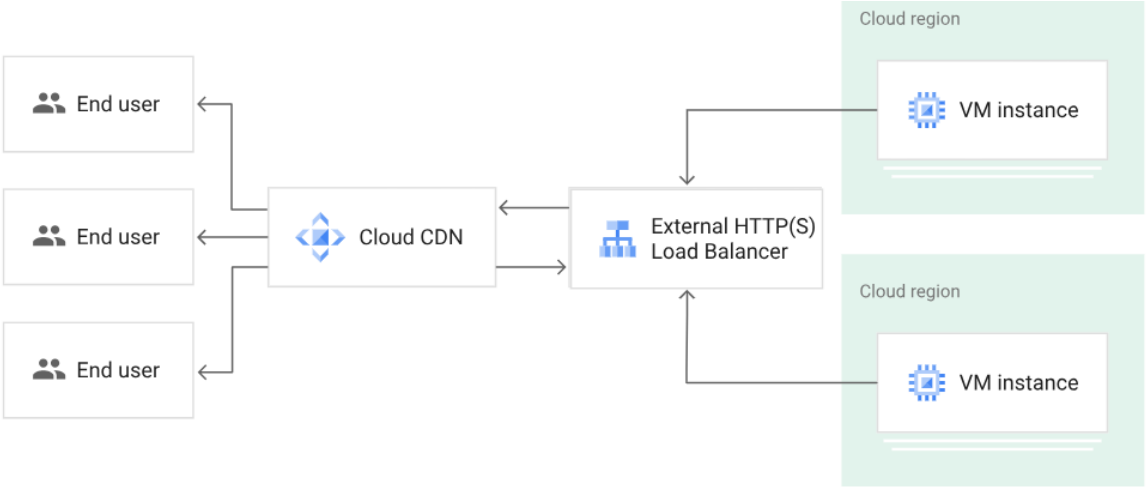
Cloud CDN works with the global external Application Load Balancer or the classic Application Load Balancer to deliver content to your users. The external Application Load Balancer provides the frontend IP addresses and ports that receive requests and the backends that respond to the requests.

Source: https://cloud.google.com/cdn/docs/overview

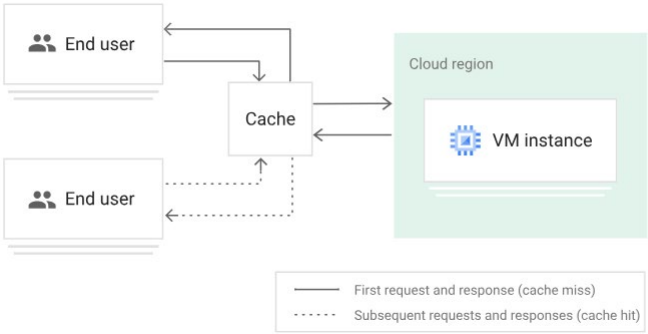
Cloud CDN content can be sourced from various types of backends.

In Cloud CDN, these backends are also called *origin servers*. Figure 1 illustrates how responses from origin servers that run on virtual machine (VM) instances flow through an external Application Load Balancer before being delivered by Cloud CDN. In this situation, the Google Front End (GFE) comprises Cloud CDN and the external Application Load Balancer.

Source: https://cloud.google.com/cdn/docs/overview



Source: https://cloud.google.com/cdn/docs/overview



Source: https://cloud.google.com/cdn/docs/overview

The first time that a piece of content is requested, the GFE determines that it can't fulfill the request from the cache. This is called a *cache miss*. When a cache miss occurs, the GFE forwards the request to the external Application Load Balancer. The load balancer then forwards the request to one of your origin servers. When the cache receives the content, the GFE forwards the content to the user.

If the origin server's response to this request is cacheable, Cloud CDN stores the response in the Cloud CDN cache for future requests. Data transfer from a cache to a client is called *cache egress*. Data transfer to a cache is called *cache fill*.

Figure 2 shows a cache hit and a cache miss:

1. Origin servers running on VM instances send HTTP(S) responses.
2. The external Application Load Balancer distributes the responses to Cloud CDN.
3. Cloud CDN delivers the responses to end users.

Source: https://cloud.google.com/cdn/docs/overview

Cache hit ratio

The *cache hit ratio* is the percentage of times that a requested object is served from the cache. If the cache hit ratio is 60%, it means that the requested object is served from the cache 60% of the time and must be retrieved from the origin 40% of the time.

For information about how cache keys can affect the cache hit ratio, see Using cache keys. For troubleshooting information, see Cache hit ratio is low.

Source: https://cloud.google.com/cdn/docs/overview

Serving content from a cache

After you enable Cloud CDN, caching happens automatically for all cacheable content. Your origin server uses HTTP headers to indicate which responses are cached. You can also control cacheability by using cache modes.

When you use a backend bucket, the origin server is Cloud Storage. When you use VM instances, the origin server is the web server software that you run on those instances.

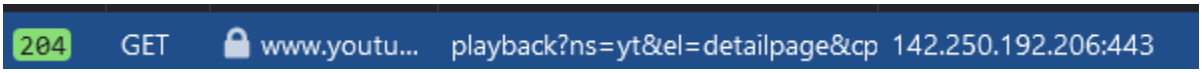
Cloud CDN uses caches in numerous locations around the world. Because of the nature of caches, it is impossible to predict whether a particular request is served out of a cache. You can, however, expect that popular requests for cacheable content are served from a cache most of the time, yielding significantly reduced latencies, reduced cost, and reduced load on your origin servers.

For more information about what Cloud CDN caches and for how long, see the Caching overview.

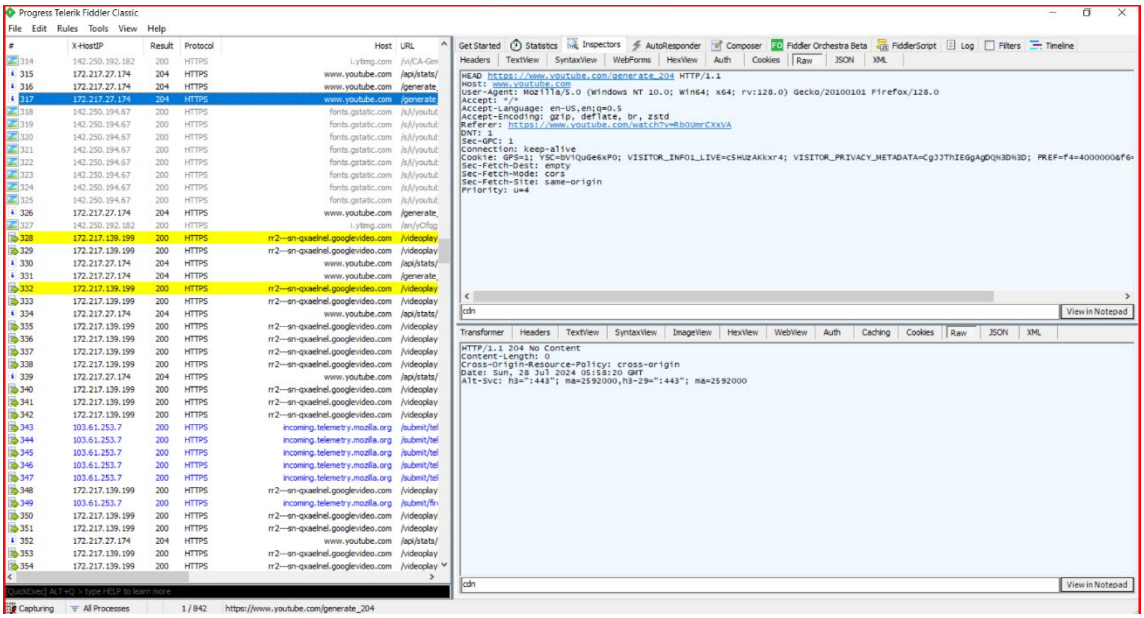
To see what Cloud CDN is serving from a cache, you can view logs.

Source: https://cloud.google.com/cdn/docs/overview

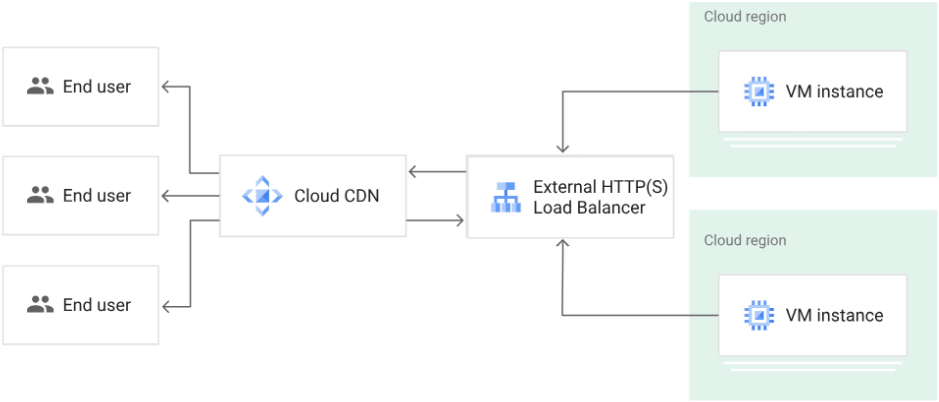
171. The '101 Accused Products include a server device for media, which is a media player. For example, Google Cloud CDN servers act as media players by hosting and streaming content via services like YouTube. They handle storage, transcoding, and real-time delivery of media. Google Cloud's infrastructure supports efficient content distribution and playback across various devices and platforms. Therefore, the Google Cloud servers with integration to its services like YouTube act as a media player. For example, the following packet capture images show that, when a request to access the content from YouTube API is placed, the packets corresponding to the streamed content are observed from Google Servers (e.g., Packet 317). It shows the content stored on Google servers can be streamed/played via API requests, and hence the Google Cloud with integration to its services like YouTube acts as a media player.



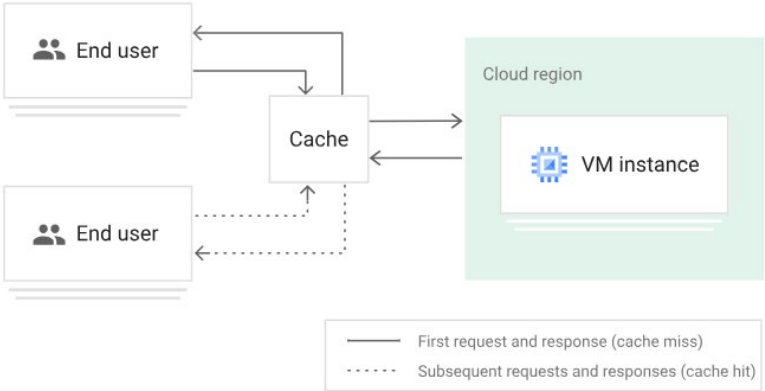
Source: Packet Capture corresponding to accessing content via YouTube



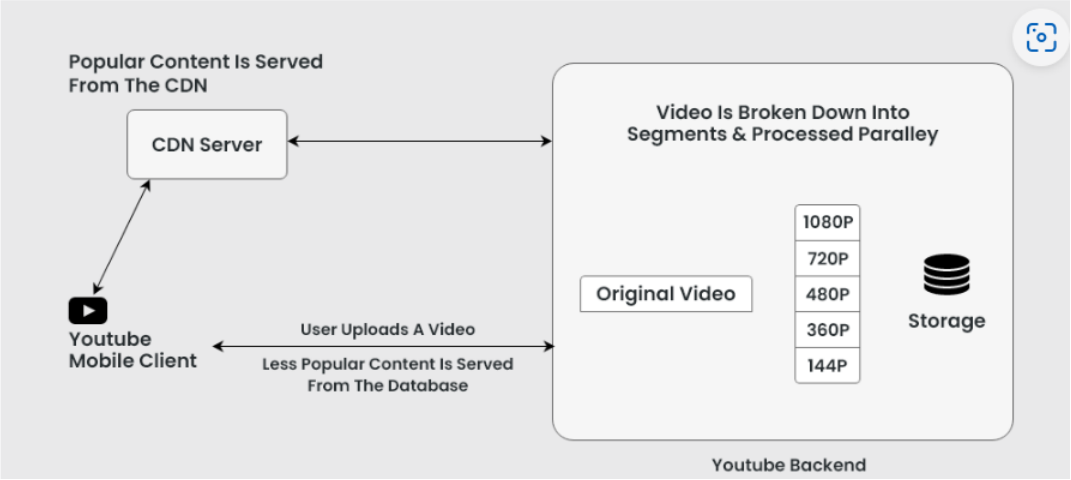
Source: You Tube Packet Capture for Accessing a user playlist



Source: https://cloud.google.com/cdn/docs/overview



Source: https://cloud.google.com/cdn/docs/overview



Source: https://www.geeksforgeeks.org/system-design-of-youtube-a-complete-architecture/

172. Google has indirectly infringed and continues to indirectly infringe one or more claims of the '101 Patent, as provided by 35 U.S.C. § 271(b), by inducing infringement by others, such as Google's customers and end-users, in this Judicial District and elsewhere in the United States. For example, Google's customers and end-users directly infringe, either literally or under the doctrine of equivalents, through their use of the inventions claimed in the '101 Patent. Google induces this direct infringement through its affirmative acts of manufacturing, selling, distributing, and/or otherwise making available the '101 Accused Products, and providing instructions, documentation, and other information to customers and end-users suggesting that they use the '101 Accused Products in an infringing manner, including technical support, marketing, product manuals, advertisements, and online documentation.[50] Because of Google's inducement, Google's customers and end-users use the '101 Accused Products in a way Google intends and they directly infringe the '101 Patent. Google performs these affirmative acts with knowledge of the '101 Patent

[50] *See, e.g.*, https://guidebooks.google.com/home; https://pixel.withgoogle.com/?hl=en&country=US; https://pixel.withgoogle.com/support/Pixel?hl=en&country=US&dark=0; https://www.youtube.com/@Googlehelp/playlists

and with the intent, or willful blindness, that the induced acts directly infringe the ’101 Patent.

173. Google has indirectly infringed and continues to indirectly infringe one or more claims of the ’101 Patent, as provided by 35 U.S.C. § 271(c), by contributing to direct infringement by others, such as customers and end-users, in this Judicial District and elsewhere in the United States. Google’s affirmative acts of selling and offering to sell the ’101 Accused Products in this Judicial District and elsewhere in the United States and causing the ’101 Accused Products to be manufactured, used, sold, and offered for sale contribute to others’ use and manufacture of the ’101 Accused Products, such that the ’101 Patent is directly infringed by others.[51] The accused components within the ’101 Accused Products including, but not limited to, software manufactured by Google, are material to the invention of the ’101 Patent, are not staple articles or commodities of commerce, have no substantial non-infringing uses, and are known by Google to be especially made or adapted for use in the infringement of the ’101 Patent. Google performs these affirmative acts with knowledge of the ’101 Patent and with intent, or willful blindness, that they cause the direct infringement of the ’101 Patent.

174. ACT has suffered damages as a result of Defendant’s direct, indirect, and willful infringement of the ’101 Patent in an amount to be proved at trial.

COUNT VI
(Infringement of the ’891 Patent)

175. Paragraphs 1 through 90 are incorporated by reference as if fully set forth herein.

176. ACT has not licensed or otherwise authorized Defendant to make, use, offer for sale, sell, or import any products that embody the inventions of the ’891 Patent.

177. Defendant has and continues to directly infringe the ’891 Patent, either literally or

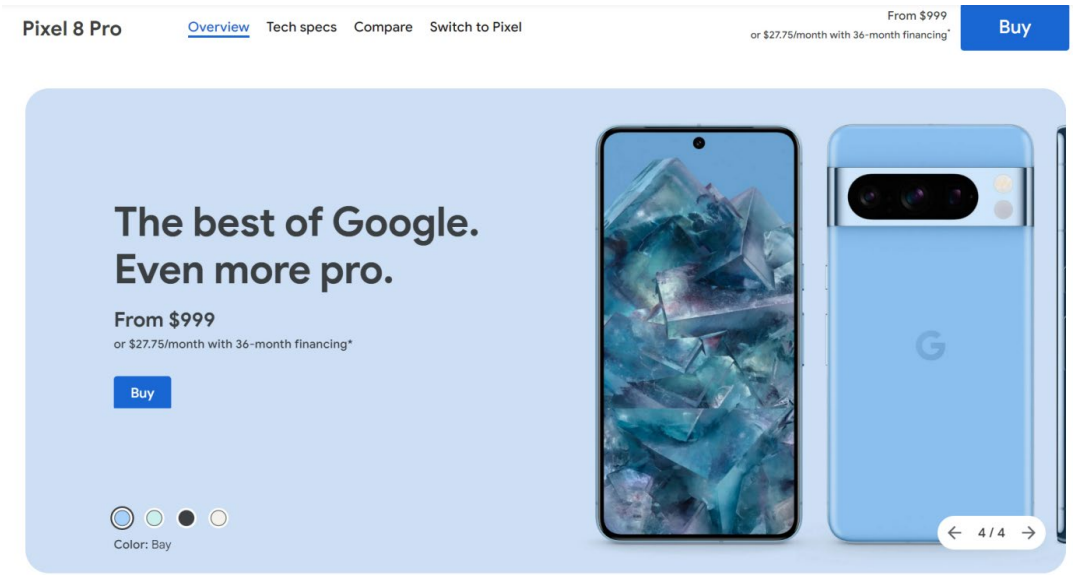
[51] *See, e.g.*, https://store.google.com/magazine/support?hl=en-US

under the doctrine of equivalents, without authority and in violation of 35 U.S.C. § 271, by making, using, offering to sell, selling, and/or importing into the United States products that satisfy each and every limitation of one or more claims of the ’891 Patent. Such products include at least Google Pixel Smartphone products compliant with 5G NR, including, but not limited to, the Pixel 4a, Pixel 5, Pixel 5a, Pixel 6, Pixel 6 Pro, Pixel 6a, Pixel 7, Pixel 7 Pro, Pixel 7a, Pixel Fold, Pixel 8, Pixel 8 Pro, and Pixel 8a (the ’891 Accused Products) which include a communication quality judging device comprising: a symbol judging means for obtaining a baseband signal representative of a sequence of multilevel symbols and judging the symbol represented by the baseband signal; a communication quality judging means for judging communication quality of a transmission channel over which the baseband signal has been transmitted, based on content of the symbol judged by the symbol judging means; and a data changing means for, if the communication quality judged by the communication quality judging means does not satisfy a predetermined condition, making a predetermined change to the data to be transmitted represented by the symbol used in the judgment, wherein at least a portion of a bit string is distinguished as a protected portion, the bit string constituting data to be transmitted represented by the sequence of symbols, and at least a portion of the symbol that belongs to the sequence of symbols contains a bit belonging to the protected portion and a redundant bit having a predetermined value, and wherein the communication quality judging means identifies the number of redundant bits having the predetermined value or the number of redundant bits missing the predetermined value among the redundant bits contained in the symbol that contains a bit belonging to the protected portion, and judges the communication quality of the transmission channel based on the identified result.

178. For example, Defendant has and continues to directly infringe at least claim 1 of the ’891 Patent by making, using, offering to sell, selling, and/or importing into the United States

the ’891 Accused Products.

179. The ’891 Accused Products are each a communication quality judging device comprising a symbol judging means for obtaining a baseband signal representative of a sequence of multilevel symbols and judging the symbol represented by the baseband signal. For example, the accused Google Pixel 8 Pro supports 5G-NR network connectivity. According to 3GPP TS 38.212, the Low-density Parity-check (LPDC) based Cyclic Redundancy Check (CRC) (i.e., a communication quality judging method) is used to detect any error in the transmitted data. CRC includes obtaining a data and control stream (i.e., a baseband signal) that consists of a plurality of bit streams (i.e., a sequence of multilevel symbols) represented as a_0, a_1…., a_{A-1} and p_0, p_1…, p_{L-1}. The CRC method further includes judging the data and control streams based on these bits (i.e., judging the symbols represented by the baseband signal).



Source: https://store.google.com/us/product/pixel_8_pro?hl=en-US

Network[22]	5G Sub 6GHz[23] Model GC3VE
	GSM/EDGE: Quad-band (850, 900, 1800, 1900 MHz)
	UMTS/HSPA+/HSDPA: Bands 1,2,4,5,6,8,19
	LTE: Bands B1/2/3/4/5/7/8/12/13/14/17/18/19/20/25/26/28/30/32/38/40/41/42/46/48/66/71
	5G Sub-6[23]: Bands n1/2/3/5/7/8/12/20/25/28/30/38/40/41/66/71/75/76/77/78
	eSIM

Source: https://store.google.com/us/product/pixel_8_pro?hl=en-US

Data and control streams from/to MAC layer are encoded /decoded to offer transport and control services over the radio transmission link. Channel coding scheme is a combination of error detection, error correcting, rate matching, interleaving and transport channel or control information mapping onto/splitting from physical channels.

Source: https://www.etsi.org/deliver/etsi_ts/138200_138299/138212/15.03.00_60/ts_138212v150300p.pdf, 9.

CRC Cyclic redundancy check

Source: https://www.etsi.org/deliver/etsi_ts/138200_138299/138212/15.03.00_60/ts_138212v150300p.pdf, 8.

7.2 Downlink shared channel and paging channel

7.2.1 Transport block CRC attachment

Error detection is provided on each transport block through a Cyclic Redundancy Check (CRC).

The entire transport block is used to calculate the CRC parity bits. Denote the bits in a transport block delivered to layer 1 by $a_0, a_1, a_2, a_3, ..., a_{A-1}$, and the parity bits by $p_0, p_1, p_2, p_3, ..., p_{L-1}$, where A is the payload size and L is the number of parity bits. The lowest order information bit a_0 is mapped to the most significant bit of the transport block as defined in Clause 6.1.1 of [TS38.321].

The parity bits are computed and attached to the DL-SCH transport block according to Clause 5.1, by setting L to 24 bits and using the generator polynomial $g_{\mathrm{CRC24A}}(D)$ if $A > 3824$; and by setting L to 16 bits and using the generator polynomial $g_{\mathrm{CRC16}}(D)$ otherwise.

The bits after CRC attachment are denoted by $b_0, b_1, b_2, b_3, ..., b_{B-1}$, where $B = A + L$.

Source: https://www.etsi.org/deliver/etsi_ts/138200_138299/138212/15.03.00_60/ts_138212v150300p.pdf, 70.

7.2.2 LDPC base graph selection

For initial transmission of a transport block with coding rate R indicated by the MCS index according to Clause 5.1.3.1 in [6, TS 38.214] and subsequent re-transmission of the same transport block, each code block of the transport block is encoded with either LDPC base graph 1 or 2 according to the following:

- if $A \leq 292$, or if $A \leq 3824$ and $R \leq 0.67$, or if $R \leq 0.25$, LDPC base graph 2 is used;
- otherwise, LDPC base graph 1 is used,

Source:
https://www.etsi.org/deliver/etsi_ts/138200_138299/138212/15.03.00_60/ts_138212v150300p.pdf, 71.

5 General procedures

Data and control streams from/to MAC layer are encoded /decoded to offer transport and control services over the radio transmission link. Channel coding scheme is a combination of error detection, error correcting, rate matching, interleaving and transport channel or control information mapping onto/splitting from physical channels.

5.1 CRC calculation

Denote the input bits to the CRC computation by $a_0, a_1, a_2, a_3, ..., a_{A-1}$, and the parity bits by $p_0, p_1, p_2, p_3, ..., p_{L-1}$, where A is the size of the input sequence and L is the number of parity bits. The parity bits are generated by one of the following cyclic generator polynomials:

- $g_{CRC24A}(D) = [D^{24} + D^{23} + D^{18} + D^{17} + D^{14} + D^{11} + D^{10} + D^7 + D^6 + D^5 + D^4 + D^3 + D + 1]$ for a CRC length $L = 24$;
- $g_{CRC24B}(D) = [D^{24} + D^{23} + D^6 + D^5 + D + 1]$ for a CRC length $L = 24$;
- $g_{CRC24C}(D) = [D^{24} + D^{23} + D^{21} + D^{20} + D^{17} + D^{15} + D^{13} + D^{12} + D^8 + D^4 + D^2 + D + 1]$ for a CRC length $L = 24$;
- $g_{CRC16}(D) = [D^{16} + D^{12} + D^5 + 1]$ for a CRC length $L = 16$;
- $g_{CRC11}(D) = [D^{11} + D^{10} + D^9 + D^5 + 1]$ for a CRC length $L = 11$;
- $g_{CRC6}(D) = [D^6 + D^5 + 1]$ for a CRC length $L = 6$.

The encoding is performed in a systematic form, which means that in GF(2), the polynomial:

$$a_0 D^{A+L-1} + a_1 D^{A+L-2} + ... + a_{A-1} D^L + p_0 D^{L-1} + p_1 D^{L-2} + ... + p_{L-2} D^1 + p_{L-1}$$

Source:
https://www.etsi.org/deliver/etsi_ts/138200_138299/138212/15.03.00_60/ts_138212v150300p.pdf, 9.

180. The ’891 Accused Products include a communication quality judging means for judging communication quality of a transmission channel over which the baseband signal has been transmitted, based on content of the symbol judged by the symbol judging means. According to 3GPP TS 38.212, the Cyclic Redundancy Check (CRC) checks for any error (i.e., judging communication quality) present in the data and control stream comprising of a plurality of bits transmitted over a radio transmission link. The error detection comprises judging the bit stream based on the payload size and parity bits, and accordingly selecting the coding rate.

5 General procedures

Data and control streams from/to MAC layer are encoded /decoded to offer transport and control services over the radio transmission link. Channel coding scheme is a combination of error detection, error correcting, rate matching, interleaving and transport channel or control information mapping onto/splitting from physical channels.

5.1 CRC calculation

Denote the input bits to the CRC computation by $a_0, a_1, a_2, a_3, ..., a_{A-1}$, and the parity bits by $p_0, p_1, p_2, p_3, ..., p_{L-1}$, where A is the size of the input sequence and L is the number of parity bits. The parity bits are generated by one of the following cyclic generator polynomials:

- $g_{CRC24A}(D) = [D^{24} + D^{23} + D^{18} + D^{17} + D^{14} + D^{11} + D^{10} + D^7 + D^6 + D^5 + D^4 + D^3 + D + 1]$ for a CRC length $L = 24$;
- $g_{CRC24B}(D) = [D^{24} + D^{23} + D^6 + D^5 + D + 1]$ for a CRC length $L = 24$;
- $g_{CRC24C}(D) = [D^{24} + D^{23} + D^{21} + D^{20} + D^{17} + D^{15} + D^{13} + D^{12} + D^8 + D^4 + D^2 + D + 1]$ for a CRC length $L = 24$;
- $g_{CRC16}(D) = [D^{16} + D^{12} + D^5 + 1]$ for a CRC length $L = 16$;
- $g_{CRC11}(D) = [D^{11} + D^{10} + D^9 + D^5 + 1]$ for a CRC length $L = 11$;
- $g_{CRC6}(D) = [D^6 + D^5 + 1]$ for a CRC length $L = 6$.

The encoding is performed in a systematic form, which means that in GF(2), the polynomial:

$$a_0 D^{A+L-1} + a_1 D^{A+L-2} + ... + a_{A-1} D^L + p_0 D^{L-1} + p_1 D^{L-2} + ... + p_{L-2} D^1 + p_{L-1}$$

Source:
https://www.etsi.org/deliver/etsi_ts/138200_138299/138212/15.03.00_60/ts_138212v150300p.pdf, 9.

7.1.3 Transport block CRC attachment

Error detection is provided on BCH transport blocks through a Cyclic Redundancy Check (CRC).

The entire transport block is used to calculate the CRC parity bits. The input bit sequence is denoted by $a'_0, a'_1, a'_2, a'_3, ..., a'_{A-1}$, and the parity bits by $p_0, p_1, p_2, p_3, ..., p_{L-1}$, where A is the payload size and L is the number of parity bits.

The parity bits are computed and attached to the BCH transport block according to Subclause 5.1 by setting L to 24 bits and using the generator polynomial $g_{CRC24C}(D)$, resulting in the sequence $b_0, b_1, b_2, b_3, ..., b_{B-1}$, where $B = A + L$.

The bit sequence $b_0, b_1, b_2, b_3, ..., b_{B-1}$ is the input bit sequence $c_0, c_1, c_2, c_3, ..., c_{K-1}$ to the channel encoder, where $c_i = b_i$ for $i = 0, 1, ..., B-1$ and $K = B$.

Source:
https://www.etsi.org/deliver/etsi_ts/138200_138299/138212/15.03.00_60/ts_138212v150300p.pdf, 70.

7.2.2 LDPC base graph selection

For initial transmission of a transport block with coding rate R indicated by the MCS index according to Clause 5.1.3.1 in [6, TS 38.214] and subsequent re-transmission of the same transport block, each code block of the transport block is encoded with either LDPC base graph 1 or 2 according to the following:

- if $A \le 292$, or if $A \le 3824$ and $R \le 0.67$, or if $R \le 0.25$, LDPC base graph 2 is used;
- otherwise, LDPC base graph 1 is used,

Source:
https://www.etsi.org/deliver/etsi_ts/138200_138299/138212/15.03.00_60/ts_138212v150300p.pdf, 71.

181. The ’891 Accused Products include a data changing means for, if the communication quality judged by the communication quality judging means does not satisfy a predetermined condition, making a predetermined change to the data to be transmitted represented by the symbol used in the judgment. According to 3GPP TS 38.212, the error detection provided by CRC includes judging the bit stream based on the payload size (A) and number of parity bits (L), and accordingly selecting the coding rate(R). Based on the values of A and R, a low-density parity check (LDPC) base graph is selected which is a method of coding and transmitting a message over a noisy transmission channel. For example, A ≤ 292 (i.e., predetermined condition), or if A ≤ 3824 and R ≤ 0.67, or if R ≤ 0.25, LDPC base graph 2 is used. Otherwise, LDPC base graph 1 is used. LDPC codes are used to correct the errors, and, therefore, correcting the error corresponds to changing data.

5 General procedures

Data and control streams from/to MAC layer are encoded /decoded to offer transport and control services over the radio transmission link. Channel coding scheme is a combination of error detection, error correcting, rate matching, interleaving and transport channel or control information mapping onto/splitting from physical channels.

5.1 CRC calculation

Denote the input bits to the CRC computation by $a_0, a_1, a_2, a_3, ..., a_{A-1}$, and the parity bits by $p_0, p_1, p_2, p_3, ..., p_{L-1}$, where A is the size of the input sequence and L is the number of parity bits. The parity bits are generated by one of the following cyclic generator polynomials:

- $g_{CRC24A}(D) = [D^{24} + D^{23} + D^{18} + D^{17} + D^{14} + D^{11} + D^{10} + D^{7} + D^{6} + D^{5} + D^{4} + D^{3} + D + 1]$ for a CRC length $L = 24$;
- $g_{CRC24B}(D) = [D^{24} + D^{23} + D^{6} + D^{5} + D + 1]$ for a CRC length $L = 24$;
- $g_{CRC24C}(D) = [D^{24} + D^{23} + D^{21} + D^{20} + D^{17} + D^{15} + D^{13} + D^{12} + D^{8} + D^{4} + D^{2} + D + 1]$ for a CRC length $L = 24$;
- $g_{CRC16}(D) = [D^{16} + D^{12} + D^{5} + 1]$ for a CRC length $L = 16$;
- $g_{CRC11}(D) = [D^{11} + D^{10} + D^{9} + D^{5} + 1]$ for a CRC length $L = 11$;
- $g_{CRC6}(D) = [D^{6} + D^{5} + 1]$ for a CRC length $L = 6$.

The encoding is performed in a systematic form, which means that in GF(2), the polynomial:

$$a_0 D^{A+L-1} + a_1 D^{A+L-2} + ... + a_{A-1} D^{L} + p_0 D^{L-1} + p_1 D^{L-2} + ... + p_{L-2} D^{1} + p_{L-1}$$

Source: https://www.etsi.org/deliver/etsi_ts/138200_138299/138212/15.03.00_60/ts_138212v150300p.pdf, 9.

7.2.2 LDPC base graph selection

For initial transmission of a transport block with coding rate R indicated by the MCS index according to Clause 5.1.3.1 in [6, TS 38.214] and subsequent re-transmission of the same transport block, each code block of the transport block is encoded with either LDPC base graph 1 or 2 according to the following:

- if $A \leq 292$, or if $A \leq 3824$ and $R \leq 0.67$, or if $R \leq 0.25$, LDPC base graph 2 is used;
- otherwise, LDPC base graph 1 is used,

Source: https://www.etsi.org/deliver/etsi_ts/138200_138299/138212/15.03.00_60/ts_138212v150300p.pdf, 70.

LDPC

LDPC stands for "Low-Density Parity-Check". It's a method used in communication systems for encoding and decoding data to detect and correct errors. LDPC codes are a type of linear error-correcting code that has found extensive application in modern communication technologies. LDPC codes are a powerful and efficient method for error correction in digital communication systems, enabling reliable data transmission even in challenging and noisy environments.

Here are some key points about LDPC:

- **Error Correction**: LDPC codes are particularly effective for correcting data transmission errors in noisy communication channels. They are used in scenarios where accurate data transmission is critical, such as in satellite communication, wireless networks, and data storage.
- **Sparse Parity-Check Matrix**: The 'low-density' in LDPC refers to the parity-check matrix used in these codes, which has a relatively small number of non-zero elements. This sparsity is what makes LDPC codes computationally efficient for error correction.

Source: https://www.sharetechnote.com/html/5G/5G_LDPC.html.

182. The ’891 Accused Products include wherein at least a portion of a bit string is distinguished as a protected portion, the bit string constituting data to be transmitted represented by the sequence of symbols, and at least a portion of the symbol that belongs to the sequence of symbols contains a bit belonging to the protected portion and a redundant bit having a predetermined value. According to 3GPP TS 38.212, each code block of the transport block is encoded with either LDPC base graph 1 or 2 based on the payload value (A) and coding rate (R) as shown below. The entire transport block is used to calculate the CRC parity bits using the generator polynomial which depends upon the number of the parity bits and payload size. The coded bit stream (i.e., a bit string) with CRC bits attached to it, is then divided into multiple blocks comprising of useful bits (i.e., a protected portion) and redundant bits represented as Cr0, Cr1…, Cr(kr-1) where r is the code block number and Kr is the number of bits for code block number r. Thus, each code segment has some useful bits and some parity/CRC bits.

7.2.2 LDPC base graph selection

For initial transmission of a transport block with coding rate R indicated by the MCS index according to Clause 5.1.3.1 in [6, TS 38.214] and subsequent re-transmission of the same transport block, each code block of the transport block is encoded with either LDPC base graph 1 or 2 according to the following:

- if $A \leq 292$, or if $A \leq 3824$ and $R \leq 0.67$, or if $R \leq 0.25$, LDPC base graph 2 is used;
- otherwise, LDPC base graph 1 is used,

Source:
https://www.etsi.org/deliver/etsi_ts/138200_138299/138212/15.03.00_60/ts_138212v150300p.pdf, 71.

6.2.3 Code block segmentation and code block CRC attachment

The bits input to the code block segmentation are denoted by $b_0, b_1, b_2, b_3, ..., b_{B-1}$ where B is the number of bits in the transport block (including CRC).

Code block segmentation and code block CRC attachment are performed according to Subclause 5.2.2.

The bits after code block segmentation are denoted by $c_{r0}, c_{r1}, c_{r2}, c_{r3}, ..., c_{r(K_r-1)}$, where r is the code block number and K_r is the number of bits for code block number r according to Subclause 5.2.2.

Source:
https://www.etsi.org/deliver/etsi_ts/138200_138299/138212/15.03.00_60/ts_138212v150300p.pdf, 71.

5G Modulation and Coding Scheme

For any communication technology, Modulation and Coding Scheme (MCS) defines the numbers of useful bits which can carried by one symbol. In contrast with 5G or 4G, a symbol is defined as Resource Element **(RE)** and MCS defined as how many useful bits can be transmitted per Resource Element (RE) . **MCS** depends on radio signal quality in wireless link, better quality the higher MCS and the more useful bits can be transmitted with in a symbol and bad signal quality result in lower MCS means less useful data can be transmitted with in a symbol.

In other words, we can say MCS depends Blocker Error Rate **(BLER)**. Typically there is a BLER threshold defined that equal to 10%. To maintain BLER not more than this value in varying radio condition Modulation and Coding Scheme (MCS) is allocated by **gNB** using link adaptation algorithm. The allocated **MCS** is signalled to the UE using DCI over PDCCH channel e.g. **DCI 1_0**, **DCI 1_1**

A **MCS** basically defines the following two aspects:

- **Modulation**
- **Code rate**

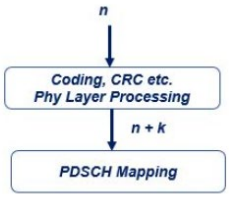
Code Rate

Code rate can be defined as the **ratio** between useful bit and total transmitted bit (Useful + Redundant Bits). These Redundant bits are added for Forward Error Correction **(FEC)**. In other words we can it is the ratio between the number of information bits at the top of the Physical layer and the number of bits which are mapped to **PDSCH** at the bottom of the **Physical layer**. We can also say, it a measure of the redundancy which is added by the Physical layer. A low coding rate corresponds to increased **redundancy**.

$$\text{Code Rate } (R) = \frac{n}{n+k}$$

$$\text{Code Rate } (R) = \frac{120}{1024} = 0.117$$

MCS Index I_{MCS}	Modulation Order Q_m	Target code Rate R x [1024]	Spectral efficiency
0	2	120	0.2344
1	2	157	0.3066
2	2	193	0.3770
3	2	251	0.4902
4	2	308	0.6016
5	2	379	0.7402



Source: https://www.techplayon.com/5g-nr-modulation-and-coding-scheme-modulation-and-code-rate/

183. The ’891 Accused Products include wherein the communication quality judging means identifies the number of redundant bits having the predetermined value or the number of redundant bits missing the predetermined value among the redundant bits contained in the symbol that contains a bit belonging to the protected portion, and judges the communication quality of the transmission channel based on the identified result. According to 3GPP TS 38.212, the bit stream is divided into multiple segments with each segment comprised of useful bits and redundant bits. Further, CRC calculations are done using the generator polynomial shown below. During the calculations, in the context of the applied input bits and a generator polynomial, redundant bits are predetermined bits. These attached CRC bits can be used for purpose error detection, which is representative of communication quality. Also, the amount of these redundant bits indicates the measure of redundancy which in turn indicates the corresponding coding rate. A low coding rate corresponds to increased redundancy and subsequent re-transmission. Hence, the channel quality can be judged by the CRC check using the generator polynomial. The communication quality judging step is done on the basis of the redundant bits contained in the symbol that contains a protected portion. The protected data is the voice data with some special features, such as sound

pressure, pitch, etc., and has the predetermined value "1". Therefore, the entire voice data coded by LDPC in the 5G NR case must include protected data. In LDPC-based coding, the code block segments are created before applying the CRC calculation and detecting the error, and therfore the code block segment contains that protected data (i.e., ...the redundant bits contained in the symbol that contains a bit belonging to the protected portion").

5 General procedures

Data and control streams from/to MAC layer are encoded /decoded to offer transport and control services over the radio transmission link. Channel coding scheme is a combination of error detection, error correcting, rate matching, interleaving and transport channel or control information mapping onto/splitting from physical channels.

5.1 CRC calculation

Denote the input bits to the CRC computation by $a_0, a_1, a_2, a_3, ..., a_{A-1}$, and the parity bits by $p_0, p_1, p_2, p_3, ..., p_{L-1}$, where A is the size of the input sequence and L is the number of parity bits. The parity bits are generated by one of the following cyclic generator polynomials:

- $g_{CRC24A}(D) = [D^{24} + D^{23} + D^{18} + D^{17} + D^{14} + D^{11} + D^{10} + D^7 + D^6 + D^5 + D^4 + D^3 + D + 1]$ for a CRC length $L = 24$;
- $g_{CRC24B}(D) = [D^{24} + D^{23} + D^6 + D^5 + D + 1]$ for a CRC length $L = 24$;
- $g_{CRC24C}(D) = [D^{24} + D^{23} + D^{21} + D^{20} + D^{17} + D^{15} + D^{13} + D^{12} + D^8 + D^4 + D^2 + D + 1]$ for a CRC length $L = 24$;
- $g_{CRC16}(D) = [D^{16} + D^{12} + D^5 + 1]$ for a CRC length $L = 16$;
- $g_{CRC11}(D) = [D^{11} + D^{10} + D^9 + D^5 + 1]$ for a CRC length $L = 11$;
- $g_{CRC6}(D) = [D^6 + D^5 + 1]$ for a CRC length $L = 6$.

The encoding is performed in a systematic form, which means that in GF(2), the polynomial:

$$a_0 D^{A+L-1} + a_1 D^{A+L-2} + ... + a_{A-1} D^L + p_0 D^{L-1} + p_1 D^{L-2} + ... + p_{L-2} D^1 + p_{L-1}$$

Source:
https://www.etsi.org/deliver/etsi_ts/138200_138299/138212/15.03.00_60/ts_138212v150300p.pdf, 9.

7.2.2 LDPC base graph selection

For initial transmission of a transport block with coding rate R indicated by the MCS index according to Clause 5.1.3.1 in [6, TS 38.214] and subsequent re-transmission of the same transport block, each code block of the transport block is encoded with either LDPC base graph 1 or 2 according to the following:

- if $A \le 292$, or if $A \le 3824$ and $R \le 0.67$, or if $R \le 0.25$, LDPC base graph 2 is used;
- otherwise, LDPC base graph 1 is used,

Source:
https://www.etsi.org/deliver/etsi_ts/138200_138299/138212/15.03.00_60/ts_138212v150300p.pdf, 71.

6.2.3 Code block segmentation and code block CRC attachment

The bits input to the code block segmentation are denoted by $b_0, b_1, b_2, b_3, ..., b_{B-1}$ where B is the number of bits in the transport block (including CRC).

Code block segmentation and code block CRC attachment are performed according to Subclause 5.2.2.

The bits after code block segmentation are denoted by $c_{r0}, c_{r1}, c_{r2}, c_{r3}, ..., c_{r(K_r-1)}$, where r is the code block number and K_r is the number of bits for code block number r according to Subclause 5.2.2.

Source: https://www.etsi.org/deliver/etsi_ts/138200_138299/138212/15.03.00_60/ts_138212v150300p.pdf, 71.

5G Modulation and Coding Scheme

For any communication technology, Modulation and Coding Scheme (MCS) defines the numbers of useful bits which can carried by one symbol. In contrast with 5G or 4G, a symbol is defined as Resource Element **(RE)** and MCS defined as how many useful bits can be transmitted per Resource Element (RE) . **MCS** depends on radio signal quality in wireless link, better quality the higher MCS and the more useful bits can be transmitted with in a symbol and bad signal quality result in lower MCS means less useful data can be transmitted with in a symbol.

In other words, we can say MCS depends Blocker Error Rate **(BLER)**. Typically there is a BLER threshold defined that equal to 10%. To maintain BLER not more than this value in varying radio condition Modulation and Coding Scheme (MCS) is allocated by **gNB** using link adaptation algorithm. The allocated **MCS** is signalled to the UE using DCI over PDCCH channel e.g. **DCI 1_0**, **DCI 1_1**

A **MCS** basically defines the following two aspects:

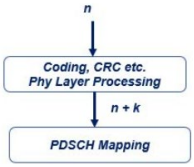
- **Modulation**
- **Code rate**

Code Rate

Code rate can be defined as the **ratio** between useful bit and total transmitted bit (Useful + Redundant Bits). These Redundant bits are added for Forward Error Correction **(FEC)**. In other words we can it is the ratio between the number of information bits at the top of the Physical layer and the number of bits which are mapped to **PDSCH** at the bottom of the **Physical layer**. We can also say, it a measure of the redundancy which is added by the Physical layer. A low coding rate corresponds to increased **redundancy**.

$$\text{Code Rate } (R) = \frac{n}{n+k}$$

$$\text{Code Rate } (R) = \frac{120}{1024} = 0.117$$



MCS Index I_{MCS}	Modulation Order Q_m	Target code Rate R x [1024]	Spectral efficiency
0	2	120	0.2344
1	2	157	0.3066
2	2	193	0.3770
3	2	251	0.4902
4	2	308	0.6016
5	2	379	0.7402

Source: https://www.techplayon.com/5g-nr-modulation-and-coding-scheme-modulation-and-code-rate/.

184. Google has indirectly infringed and continues to indirectly infringe one or more claims of the ’891 Patent, as provided by 35 U.S.C. § 271(b), by inducing infringement by others,

such as Google's customers and end-users, in this Judicial District and elsewhere in the United States. For example, Google's customers and end-users directly infringe, either literally or under the doctrine of equivalents, through their use of the inventions claimed in the '891 Patent. Google induces this direct infringement through its affirmative acts of manufacturing, selling, distributing, and/or otherwise making available the '891 Accused Products, and providing instructions, documentation, and other information to customers and end-users suggesting that they use the '891 Accused Products in an infringing manner, including technical support, marketing, product manuals, advertisements, and online documentation.[52] Because of Google's inducement, Google's customers and end-users use the '891 Accused Products in a way Google intends and they directly infringe the '891 Patent. Google performs these affirmative acts with knowledge of the '891 Patent and with the intent, or willful blindness, that the induced acts directly infringe the '891 Patent.

185. Google has indirectly infringed and continues to indirectly infringe one or more claims of the '891 Patent, as provided by 35 U.S.C. § 271(c), by contributing to direct infringement by others, such as customers and end-users, in this Judicial District and elsewhere in the United States. Google's affirmative acts of selling and offering to sell the '891 Accused Products in this Judicial District and elsewhere in the United States and causing the '891 Accused Products to be manufactured, used, sold, and offered for sale contribute to others' use and manufacture of the '891 Accused Products, such that the '891 Patent is directly infringed by others.[53] The accused components within the '891 Accused Products including, but not limited to, software manufactured by Google, are material to the invention of the '891 Patent, are not staple articles or

---

[52] *See, e.g.*, https://guidebooks.google.com/home; https://pixel.withgoogle.com/?hl=en&country=US; https://pixel.withgoogle.com/support/Pixel?hl=en&country=US&dark=0; https://www.youtube.com/@Googlehelp/playlists

[53] *See, e.g.*, https://store.google.com/magazine/support?hl=en-US

commodities of commerce, have no substantial non-infringing uses, and are known by Google to be especially made or adapted for use in the infringement of the ’891 Patent. Google performs these affirmative acts with knowledge of the ’891 Patent and with intent, or willful blindness, that they cause the direct infringement of the ’891 Patent.

186. ACT has suffered damages as a result of Defendant’s direct and indirect infringement of the ’891 Patent in an amount to be proved at trial.

DEMAND FOR JURY TRIAL

Plaintiff hereby demands a jury for all issues so triable.

PRAYER FOR RELIEF

WHEREFORE, ACT prays for relief against Defendant as follows:

a. Entry of judgment declaring that Defendant has directly and/or indirectly infringed one or more claims of each of the Patents-in-Suit;

b. An order pursuant to 35 U.S.C. § 283 permanently enjoining Defendant, their officers, agents, servants, employees, attorneys, and those persons in active concert or participation with them, from further acts of infringement of the Patents-in-Suit;

c. An order awarding damages sufficient to compensate ACT for Defendant’s infringement of the Patents-in-Suit, but in no event less than a reasonable royalty, together with interest and costs;

d. Entry of judgment declaring that Defendant’s infringement has been willful and awarding ACT treble damages pursuant to 35 U.S.C. § 284; and

e. Entry of judgment declaring that this case is exceptional and awarding ACT its costs and reasonable attorney fees under 35 U.S.C. § 285; and

f. Such other and further relief as the Court deems just and proper.

Dated: August 2, 2024

Respectfully submitted,

/s/ Peter Lambrianakos
Alfred R. Fabricant
NY Bar No. 2219392
Email: ffabricant@fabricantllp.com
Peter Lambrianakos
NY Bar No. 2894392
Email: plambrianakos@fabricantllp.com
Vincent J. Rubino, III
NY Bar No. 4557435
Email: vrubino@fabricantllp.com
Joseph M. Mercadante
NY Bar No. 4784930
Email: jmercadante@fabricantllp.com
FABRICANT LLP
411 Theodore Fremd Avenue, Suite 206 South
Rye, New York 10580
Telephone: (212) 257-5797
Facsimile: (212) 257-5796

Samuel F. Baxter
Texas Bar No. 01938000
sbaxter@mckoolsmith.com
Jennifer L. Truelove
Texas Bar No. 24012906
jtruelove@mckoolsmith.com
MCKOOL SMITH, P.C.
104 E. Houston Street, Suite 300
Marshall, Texas 75670
Telephone: (903) 923-9000
Facsimile: (903) 923-9099

ATTORNEYS FOR PLAINTIFF
ADVANCED CODING TECHNOLOGIES, LLC