

**UNITED STATES DISTRICT COURT
FOR THE WESTERN DISTRICT OF TEXAS
WACO DIVISION**

WAG ACQUISITION, L.L.C.,

Plaintiff,

v.

GOOGLE LLC and
YOUTUBE, INC.,

Defendants.

§
§
§
§
§
§
§
§
§
§

No. 6:21-cv-00816-ADA

Jury Trial Demanded

WAG ACQUISITION, L.L.C.’S PRELIMINARY INFRINGEMENT CONTENTIONS

Pursuant to the Honorable Judge Albright’s Standing Order Governing Proceedings – Patent Cases (Dkt. No. 16 ¶ 2), Plaintiff WAG Acquisition, L.L.C., through the undersigned counsel, hereby serves its preliminary infringement contentions on Defendants Google LLC and YouTube, Inc.¹ (“Defendants”).

The attached charts for U.S. Patent Nos. 9,742,824; 9,729,594; and 9,762,636 lay out the claims Defendants are accused of infringing and set forth where in the accused products each element of the asserted claims is found.

All claims asserted herein are entitled to the priority date (*i.e.*, the earliest date of invention) of November 11, 1999.

Plaintiff is producing, concurrent herewith, documents evidencing conception and reduction to practice for each claimed invention and a copy of the file history for each of the

¹ Plaintiff understands that, per Defendants’ Rule 7.1 Statement, YouTube, Inc. should be listed as YouTube LLC. Dkt. No. 18 at 1 n.1.

patents-in-suit. Plaintiff has made a good faith effort to produce documents evidencing conception and reduction to practice. Should Plaintiff identify additional relevant documents, Plaintiff will supplement this production.

Plaintiff reserves the right to amend these contentions in light of discovery, which has not yet begun, as well as to take into account any claim construction rulings that the Court may issue.

Plaintiff has marked the accompanying claim charts as Confidential pursuant to the interim Protective Order issued in this case. Dkt. No. 16 at 3.

HALEY & OLSON, P.C.

100 North Ritchie Road, Suite 200

Waco, Texas 76712

Tel: (254) 776-3336

Fax: (254) 776-6823

By: s/ Brandon R. Oates

Brandon R. Oates (State Bar No. 24032921)

Email: boates@haleyolson.com

OF COUNSEL:

LISTON ABRAMSON LLP

The Chrysler Building

405 Lexington Ave, 46th Floor

New York, New York 10174

Tel: (212) 257-1630

Ronald Abramson (*Admitted pro hac vice*)

David G. Liston (*Admitted pro hac vice*)

Ari J. Jaffess (*Admitted pro hac vice*)

Alex G. Patchen (*Admitted pro hac vice*)

M. Michael Lewis (*Admitted pro hac vice*)

Gina K. Kim

Email: docket@listonabramson.com

Attorneys for Plaintiff WAG Acquisition, L.L.C.

CERTIFICATE OF SERVICE

I, Ronald Abramson, hereby certify that on this day a true and correct copy of the foregoing document, WAG Acquisition, L.L.C.'s Preliminary Infringement Contentions, was served on all attorneys of record for Defendants via electronic mail.

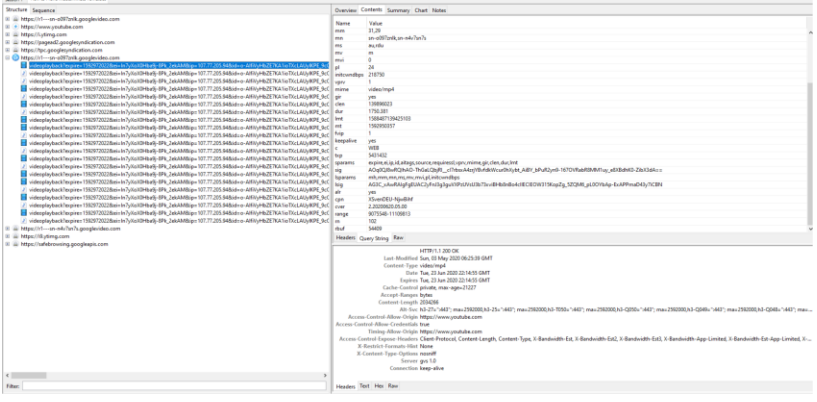
Dated: November 15, 2021

/s/ Ronald Abramson

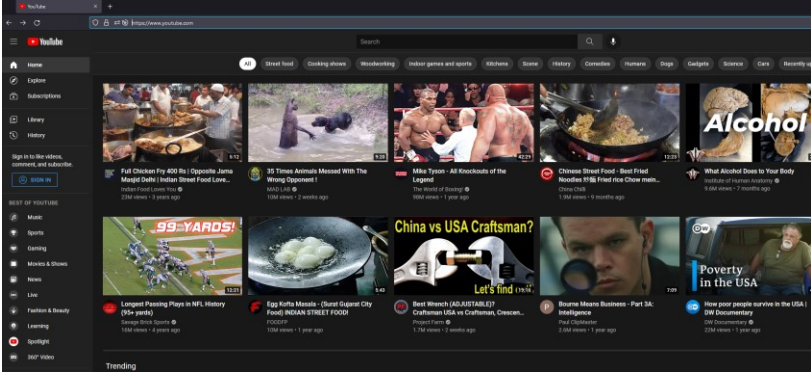
Ronald Abramson

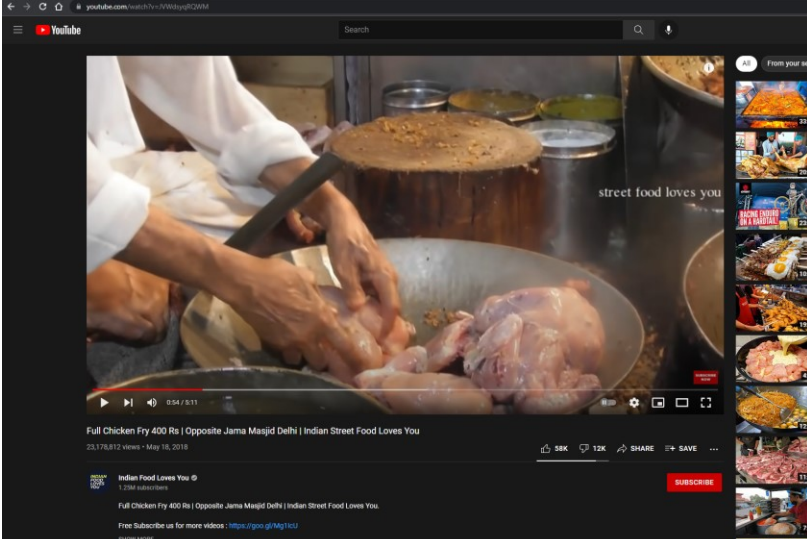
Claim Language	Contention
Overview	WAG has asserted three patents against YouTube LLC and/or affiliated entities such as Google LLC (“YouTube”), which concern streaming media over the Internet, conducted between a user (client) system, which requests and plays the media, and a server to which the client system connects over the Internet, which provides the requested media. WAG’s ’824 patent (the subject of the within contentions) concerns methods, apparatus, and computer program products on the server side of this interaction, for prerecorded media. WAG’s ’594 patent concerns methods, apparatus, and computer program products on the client side of the above-described interaction. WAG’s ’636 patent corresponds to the ’824 patent in certain respects, but concerns live media (e.g., live performances) created contemporaneously with its streaming. Because the subject matter of these patents overlap, WAG’s infringement contentions with regard to the ’594 and ’636 patents are each incorporated herein by reference. YouTube directs and controls the activity of streaming digitally encoded media it provides over the Internet through the site youtube.com (also accessible as www.youtube.com) (the “YouTube Web Site”) from servers owned and/or the operations of which are controlled by YouTube, including servers in the U.S. and in the Western District of Texas (“YouTube’s Servers”). These contentions relate to every streaming video program accessible from or through the YouTube Site for the period from the 2017 issuance dates of the patent addressed in these contentions, through the common date of expiration of the patent (March 28, 2021). They further relate to videos streamed from YouTube’s Servers (e.g., googlevideo.com, tv.youtube.com) to embedded “YouTube” devices and apps. Also included in these contentions is streaming audio distribution from music.youtube.com (formerly Google Play Music), as well as the YouTube TV subscription service (available for distribution through tv.youtube.com). The foregoing services are addressed individually, but are also referred to, for purposes of collective reference, as the “YouTube Streaming Services.” YouTube distributes pre-recorded video programs from YouTube’s Servers, over the Internet, to large numbers of users in the U.S. and elsewhere. During the relevant period, this was done via a pull protocol defined at the application layer and generally conforming with the MPEG-DASH specification. (Google is a Charter Member of the DASH Industry Forum.) YouTube’s web pages also contain reference to a manifest for HTTP Live Streaming (“HLS”), in addition to providing a link to an MPEG-DASH manifest. HLS is a streaming media delivery specification, specified in IETF RFC 8216. Systems in

Claim Language	Contention
	the U.S. that distribute streaming media in accordance with the HLS specification infringe WAG's patents asserted herein on substantially the same facts shown herein with respect to YouTube's use of MPEG-DASH. The client-side JavaScript files that YouTube distributes in support of its streaming media (e.g., base.js) also provide functions for HLS and YouTube web pages provide links to HLS manifests as well as MPEG-DASH manifests. WAG contends that YouTube has supported streaming media delivery during the relevant period via HLS as at least a fallback where the user device identifies itself as a device that normally will not support MPEG-DASH or prefers HLS. YouTube has used its own Google Cloud CDN (which includes edge servers known as Google Global Cache ("GGC") servers at locations including Midland, El Paso, Austin, and San Antonio, Texas), and also partnered with third-party providers, including without limitation Fastly, Cloudflare, Highwinds, Level 3, and Akamai, for CDN distribution. The pull protocol employed by YouTube is based on HTTP (or HTTPS) requests, which can be responded to through CDN servers at the network Edge that support HTTP/HTTPS retrieval (as most ordinary web servers do), provided the edge server possesses the relevant media data elements. The same streaming mechanisms appear to be consistently supported through each CDN, independent of the type of user-agent on the client side, and in a similar manner over the respective CDNs. The first example shown below, from June 2020, is for a client-side device comprising a Chrome browser on a desktop computer. Insofar as relevant to meeting the limitations below, the evidence is the same in all material respects for all widely used browsers and devices (Chrome, Firefox, Internet Explorer, Edge, Safari, etc.) iOS, iPadOS, Android, FireOS, FireTV, Roku, TVOS and other "Smart TV" user systems, regardless of the CDN. The MPEG-DASH form of distribution is reflected in data such as the following, observed through a web debugging proxy inserted in the communication path between a user system accessing a YouTube prerecorded video, and the YouTube Server (at a server in the domain googlevideo.com) (The proxy provided is solely for data acquisition and testing purposes and is not in any way necessary to use YouTube's Streaming Services, nor does it change any of the requests or responses while viewing video). The same request-response patterns may also be readily observed, on desktop and other non-embedded platforms supporting full-featured browsers, through the "developer tools" and like utility functions built into

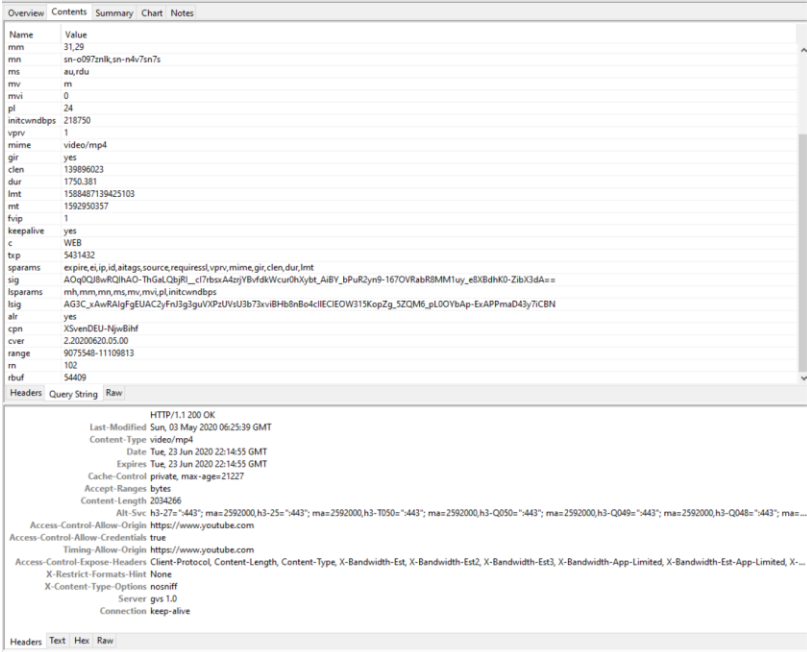
Claim Language	Contention
	the browsers, for example, by hitting Ctrl-Shift-I and looking at the Network tab during playback. The following is a representative example of the requests and responses observed (in this sample, taken in June 2020 using a proxy) in streaming a prerecorded video program from YouTube:  The entirety of each requested video program (from beginning to end) is delivered as a series of time-sequenced media data elements, where each such element has been given one or more serial identifiers (e.g., range=9075548-11109813, rn=102), and requested by serial identifier, as reflected by the sample sequence above, reflecting 11 such successive requests, interleaved with similarly sequenced requests for the audio portion of the program. The client-side software in these examples is responsible for requesting each such element, by its serial identifier, when and as needed by the client, in order to sustain continuous and uninterrupted playback. Because the streaming media flow as to each element in the stream is thus dependent on specific client-side requests for the respective elements comprising the stream, the overall process is sometimes referred to as a “pull.” Furthermore, in order for the client to successfully keep up with the stream while making such pull requests, it should not take longer to request and receive the individual segments than the time it takes to play the segments back at a normal rendition. Repeated observations confirm that transmissions from YouTube Servers consistently meet these requirements. For example, in the below sequence, requesting and receiving video data having 3m 44 sec of playback time only uses 1m 59sec. for the transfer, meaning that the data is transferred in about half the time it takes for the client system to play it back:

Claim Language	Contention																																																																																																			
	Structure: Sequence <table><tr><th>Code</th><th>Met...</th><th>Host</th><th>Path</th><th>Start</th><th>Duration</th><th>Size</th><th>Status</th><th>Info</th></tr><tr><td>200</td><td>GET</td><td>r1---sn-o087mnl...</td><td>/videoplayback?expire=1502972022&ei=ln7yloX0H4uylj-SPk_2ekAM8Iq=107.77.205.94&id=o-AWlYh6ZETKA1oTicLA...</td><td>15:14:51</td><td>12.15 s</td><td>1.97 MB</td><td>Complete</td><td></td></tr><tr><td>200</td><td>GET</td><td>r1---sn-o087mnl...</td><td>/videoplayback?expire=1502972022&ei=ln7yloX0H4uylj-SPk_2ekAM8Iq=107.77.205.94&id=o-AWlYh6ZETKA1oTicLA...</td><td>15:15:10</td><td>10.35 s</td><td>1.62 MB</td><td>Complete</td><td></td></tr><tr><td>200</td><td>GET</td><td>r1---sn-o087mnl...</td><td>/videoplayback?expire=1502972022&ei=ln7yloX0H4uylj-SPk_2ekAM8Iq=107.77.205.94&id=o-AWlYh6ZETKA1oTicLA...</td><td>15:15:22</td><td>12.21 s</td><td>1.95 MB</td><td>Complete</td><td></td></tr><tr><td>200</td><td>GET</td><td>r1---sn-o087mnl...</td><td>/videoplayback?expire=1502972022&ei=ln7yloX0H4uylj-SPk_2ekAM8Iq=107.77.205.94&id=o-AWlYh6ZETKA1oTicLA...</td><td>15:15:37</td><td>11.92 s</td><td>1.97 MB</td><td>Complete</td><td></td></tr><tr><td>200</td><td>GET</td><td>r1---sn-o087mnl...</td><td>/videoplayback?expire=1502972022&ei=ln7yloX0H4uylj-SPk_2ekAM8Iq=107.77.205.94&id=o-AWlYh6ZETKA1oTicLA...</td><td>15:16:09</td><td>12.38 s</td><td>1.95 MB</td><td>Complete</td><td></td></tr><tr><td>200</td><td>GET</td><td>r1---sn-o087mnl...</td><td>/videoplayback?expire=1502972022&ei=ln7yloX0H4uylj-SPk_2ekAM8Iq=107.77.205.94&id=o-AWlYh6ZETKA1oTicLA...</td><td>15:16:35</td><td>9.78 s</td><td>1.64 MB</td><td>Complete</td><td></td></tr><tr><td>200</td><td>GET</td><td>r1---sn-o087mnl...</td><td>/videoplayback?expire=1502972022&ei=ln7yloX0H4uylj-SPk_2ekAM8Iq=107.77.205.94&id=o-AWlYh6ZETKA1oTicLA...</td><td>15:16:56</td><td>9.61 s</td><td>1.62 MB</td><td>Complete</td><td></td></tr><tr><td>200</td><td>GET</td><td>r1---sn-o087mnl...</td><td>/videoplayback?expire=1502972022&ei=ln7yloX0H4uylj-SPk_2ekAM8Iq=107.77.205.94&id=o-AWlYh6ZETKA1oTicLA...</td><td>15:17:17</td><td>9.76 s</td><td>1.64 MB</td><td>Complete</td><td></td></tr><tr><td>200</td><td>GET</td><td>r1---sn-o087mnl...</td><td>/videoplayback?expire=1502972022&ei=ln7yloX0H4uylj-SPk_2ekAM8Iq=107.77.205.94&id=o-AWlYh6ZETKA1oTicLA...</td><td>15:17:39</td><td>10.87 s</td><td>1.83 MB</td><td>Complete</td><td></td></tr><tr><td>200</td><td>GET</td><td>r1---sn-o087mnl...</td><td>/videoplayback?expire=1502972022&ei=ln7yloX0H4uylj-SPk_2ekAM8Iq=107.77.205.94&id=o-AWlYh6ZETKA1oTicLA...</td><td>15:18:05</td><td>9.61 s</td><td>1.62 MB</td><td>Complete</td><td></td></tr></table> Filter: Overview Summary Chart Name Value Requests 11 Completed 11 Incomplete 0 Failed 0 Blocked 0 DNS 0 Connects 1 TLS Handshakes 1 Kept Alive 10 Timing Start 6/23/20 15:14:51 End 6/23/20 15:18:36 TimeRange 3 m 44 s Request Latency 2.04 s Duration 1 m 59 s Min 9.61 s Mean 10.82 s Max 12.38 s Total 1 m 59 s DNS - Connect 37 ms TLS Handshake 289 ms Latency 1.58 s Speed 168.08 KB/s Request Speed 16.77 MB/s Response Speed 170.67 KB/s Size Requests 17.17 KB Responses 19.52 MB Combined 19.53 MB Compression - The MPEG-DASH standard, ISO/IEC 23009, spells out media distribution steps that align identically with each of the claim limitations herein. The version of the MPEG-DASH specification referenced herein is the Fourth edition, Dec. 2019. However, WAG believes that YouTube also complied with earlier and later versions of this standard, with no differences between such versions being material to the claims herein. The foregoing figures are representative of every platform streaming from YouTube Servers for which WAG has been able to observe the contents of HTTP(S) traffic, including Chrome, Firefox, Safari, and other desktop browsers, iOS on iPhone and iPad, and Android, as well as TVOS and FireOS. It is also representative of Google Play Music/YouTube Music, except insofar as these audio services are limited to audio media data elements (still nevertheless internally named “videoplayback”). On some embedded platforms (such as Roku and TVOS), due to encryption, WAG has been unable to inspect the contents of HTTPS (encrypted) requests and responses. However, the timing of the encrypted requests and responses follow the same pattern as observable on other platforms (e.g., desktop Chrome, as well as the FireOS embedded platform), and WAG believes the MPEG-DASH streaming implementation is the same in all material respects to the other observed implementations. WAG will include in its discovery requests inquiries directed to the additional platforms whose communications have been obscured by encryption.	Code	Met...	Host	Path	Start	Duration	Size	Status	Info	200	GET	r1---sn-o087mnl...	/videoplayback?expire=1502972022&ei=ln7yloX0H4uylj-SPk_2ekAM8Iq=107.77.205.94&id=o-AWlYh6ZETKA1oTicLA...	15:14:51	12.15 s	1.97 MB	Complete		200	GET	r1---sn-o087mnl...	/videoplayback?expire=1502972022&ei=ln7yloX0H4uylj-SPk_2ekAM8Iq=107.77.205.94&id=o-AWlYh6ZETKA1oTicLA...	15:15:10	10.35 s	1.62 MB	Complete		200	GET	r1---sn-o087mnl...	/videoplayback?expire=1502972022&ei=ln7yloX0H4uylj-SPk_2ekAM8Iq=107.77.205.94&id=o-AWlYh6ZETKA1oTicLA...	15:15:22	12.21 s	1.95 MB	Complete		200	GET	r1---sn-o087mnl...	/videoplayback?expire=1502972022&ei=ln7yloX0H4uylj-SPk_2ekAM8Iq=107.77.205.94&id=o-AWlYh6ZETKA1oTicLA...	15:15:37	11.92 s	1.97 MB	Complete		200	GET	r1---sn-o087mnl...	/videoplayback?expire=1502972022&ei=ln7yloX0H4uylj-SPk_2ekAM8Iq=107.77.205.94&id=o-AWlYh6ZETKA1oTicLA...	15:16:09	12.38 s	1.95 MB	Complete		200	GET	r1---sn-o087mnl...	/videoplayback?expire=1502972022&ei=ln7yloX0H4uylj-SPk_2ekAM8Iq=107.77.205.94&id=o-AWlYh6ZETKA1oTicLA...	15:16:35	9.78 s	1.64 MB	Complete		200	GET	r1---sn-o087mnl...	/videoplayback?expire=1502972022&ei=ln7yloX0H4uylj-SPk_2ekAM8Iq=107.77.205.94&id=o-AWlYh6ZETKA1oTicLA...	15:16:56	9.61 s	1.62 MB	Complete		200	GET	r1---sn-o087mnl...	/videoplayback?expire=1502972022&ei=ln7yloX0H4uylj-SPk_2ekAM8Iq=107.77.205.94&id=o-AWlYh6ZETKA1oTicLA...	15:17:17	9.76 s	1.64 MB	Complete		200	GET	r1---sn-o087mnl...	/videoplayback?expire=1502972022&ei=ln7yloX0H4uylj-SPk_2ekAM8Iq=107.77.205.94&id=o-AWlYh6ZETKA1oTicLA...	15:17:39	10.87 s	1.83 MB	Complete		200	GET	r1---sn-o087mnl...	/videoplayback?expire=1502972022&ei=ln7yloX0H4uylj-SPk_2ekAM8Iq=107.77.205.94&id=o-AWlYh6ZETKA1oTicLA...	15:18:05	9.61 s	1.62 MB	Complete	
Code	Met...	Host	Path	Start	Duration	Size	Status	Info																																																																																												
200	GET	r1---sn-o087mnl...	/videoplayback?expire=1502972022&ei=ln7yloX0H4uylj-SPk_2ekAM8Iq=107.77.205.94&id=o-AWlYh6ZETKA1oTicLA...	15:14:51	12.15 s	1.97 MB	Complete																																																																																													
200	GET	r1---sn-o087mnl...	/videoplayback?expire=1502972022&ei=ln7yloX0H4uylj-SPk_2ekAM8Iq=107.77.205.94&id=o-AWlYh6ZETKA1oTicLA...	15:15:10	10.35 s	1.62 MB	Complete																																																																																													
200	GET	r1---sn-o087mnl...	/videoplayback?expire=1502972022&ei=ln7yloX0H4uylj-SPk_2ekAM8Iq=107.77.205.94&id=o-AWlYh6ZETKA1oTicLA...	15:15:22	12.21 s	1.95 MB	Complete																																																																																													
200	GET	r1---sn-o087mnl...	/videoplayback?expire=1502972022&ei=ln7yloX0H4uylj-SPk_2ekAM8Iq=107.77.205.94&id=o-AWlYh6ZETKA1oTicLA...	15:15:37	11.92 s	1.97 MB	Complete																																																																																													
200	GET	r1---sn-o087mnl...	/videoplayback?expire=1502972022&ei=ln7yloX0H4uylj-SPk_2ekAM8Iq=107.77.205.94&id=o-AWlYh6ZETKA1oTicLA...	15:16:09	12.38 s	1.95 MB	Complete																																																																																													
200	GET	r1---sn-o087mnl...	/videoplayback?expire=1502972022&ei=ln7yloX0H4uylj-SPk_2ekAM8Iq=107.77.205.94&id=o-AWlYh6ZETKA1oTicLA...	15:16:35	9.78 s	1.64 MB	Complete																																																																																													
200	GET	r1---sn-o087mnl...	/videoplayback?expire=1502972022&ei=ln7yloX0H4uylj-SPk_2ekAM8Iq=107.77.205.94&id=o-AWlYh6ZETKA1oTicLA...	15:16:56	9.61 s	1.62 MB	Complete																																																																																													
200	GET	r1---sn-o087mnl...	/videoplayback?expire=1502972022&ei=ln7yloX0H4uylj-SPk_2ekAM8Iq=107.77.205.94&id=o-AWlYh6ZETKA1oTicLA...	15:17:17	9.76 s	1.64 MB	Complete																																																																																													
200	GET	r1---sn-o087mnl...	/videoplayback?expire=1502972022&ei=ln7yloX0H4uylj-SPk_2ekAM8Iq=107.77.205.94&id=o-AWlYh6ZETKA1oTicLA...	15:17:39	10.87 s	1.83 MB	Complete																																																																																													
200	GET	r1---sn-o087mnl...	/videoplayback?expire=1502972022&ei=ln7yloX0H4uylj-SPk_2ekAM8Iq=107.77.205.94&id=o-AWlYh6ZETKA1oTicLA...	15:18:05	9.61 s	1.62 MB	Complete																																																																																													

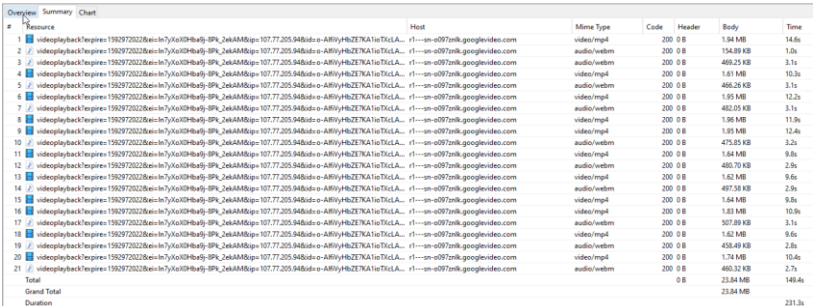
Claim Language	Contention
[1.P.1] A method	This method is performed when prerecorded streaming video programming is distributed by YouTube's Servers to users in the U.S.
[1.P.2] for distributing over the Internet,	The YouTube Streaming Services are accessible over the Internet through the youtube.com/www.youtube.com website, as well as directly from YouTube Servers including googlevideo.com and tv.youtube.com. For example:  Activating a link on youtube.com/www.youtube.com (or similarly requesting a video through a Roku app or the like) results in a transmission of the streaming media over the Internet. For example, clicking on the “Full Chicken Fry” video results in transmission of that video program. Programming on YouTube ranges from short clips of under a minute to full-length movie, sports, news, and current events programs (among others), which can have durations of two or more hours. (YouTube live programming, addressed under separate contentions, can run continuously.)
[1.P.3] from a server system	YouTube's videos are distributed from servers accessible through youtube.com/www.youtube.com, or directly from servers such as googlevideo.com and tv.youtube.com:

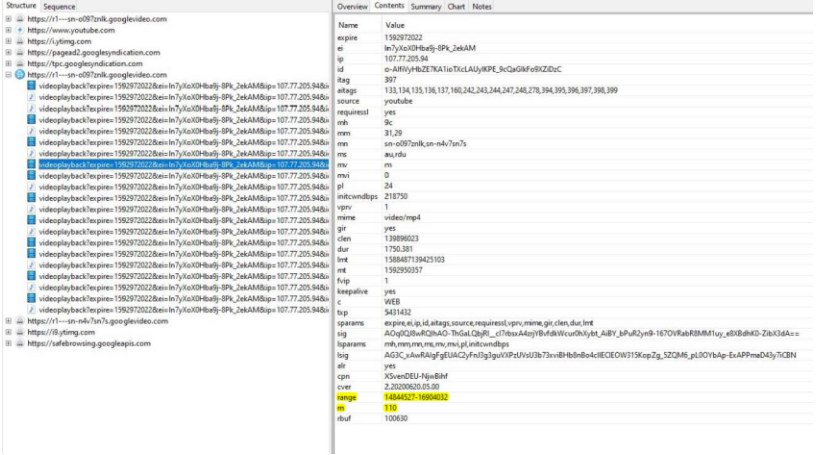
Claim Language	Contention
	
[1.P.4] to one or more user systems,	The above screen captures each reflect transmission to one user system. For any given video there are many users similarly situated and the server system streams audio/video programs to them all as recited herein.
[1.P.5] a pre-recorded audio or video program	Each of the programs listed on the YouTube home page (e.g., “Full Chicken Fry,” “35 Times Animals Messed with the Wrong Opponent,” etc., and many others) are prerecorded video programs. (There is a section lower down the page for “Live” programming, which is separately addressed.)
[1.P.6] stored in digitally encoded form on computer-readable media,	The web browser request shown above for “Full Chicken Fry,” https://www.youtube.com/watch?v=JWVdsyqRQWM reflects a request for a resource accessible to the youtube.com server. WAG does not have access to direct proof of how YouTube videos (and/or audio programs) are stored. However, a POSITA understands that the requested resource in a case such as shown here will reside in digitally encoded form on computer-readable media accessible to the server, so that the server can fulfill the user request. WAG does not believe YouTube will dispute this fact, and contends that an inference to support the fact is reasonable, whether YouTube expressly concedes the point or not, to establish that YouTube stores the pre-recorded video programs that it makes available through the YouTube Streaming Services in digitally encoded form on computer-readable media.

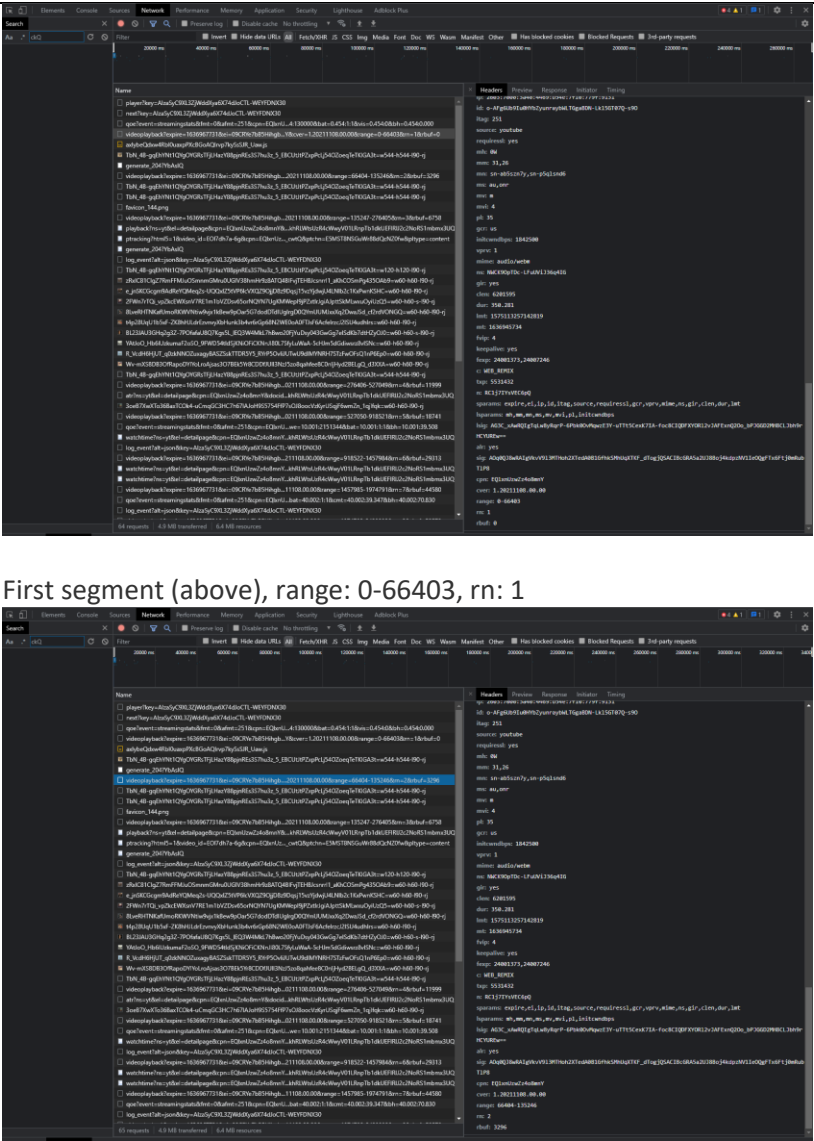
Claim Language	Contention
	Moreover, YouTube's MPEG-DASH videos are requested by the user device, by successive segments of video and audio, pursuant to a manifest supplied to the user device by the YouTube server. Such manifests contain entries such as the following, indicating storage of the successive audio and video segments in a file system (or operationally equivalent data structure) on the YouTube servers: <pre> <SegmentList> <Initialization sourceURL="range/0-258"/> <SegmentURL media="range/460-187101"/> <SegmentURL media="range/187102-376159"/> <SegmentURL media="range/376160-561636"/> <SegmentURL media="range/561637-745450"/> <SegmentURL media="range/745451-926622"/> <SegmentURL media="range/926623-1114800"/> <SegmentURL media="range/1114801-1299024"/> <SegmentURL media="range/1299025-1481544"/> <SegmentURL media="range/1481545-1664235"/> <SegmentURL media="range/1664236-1844136"/> <SegmentURL media="range/1844137-2020117"/> <SegmentURL media="range/2020118-2151945"/> </SegmentList> </pre> The above is from the manifest for the video at https://www.youtube.com/watch?v=i2lhwb_OckQ (a different video, which also reflects a series of requests/responses by range and rn as shown in the first example above). This manifest is designed, per the MPEG-DASH specification, to cause the user device to sequentially request the listed media segments, resulting in the continuous playback of the subject video. It is reasonable to infer that the right side of each of the above expressions represents a path to a storage location of the corresponding segment, on the YouTube server, wherein "range" represents a directory (folder), and "[number]-[number]" represents a file (or an equivalent to a folder-file arrangement, as implemented in a database or other data structure). Even in such cases (such as with third-party CDNs) where YouTube does not actually own the distributing server, it nevertheless has administrative control over its operation, and thus directs the storage on the server of the digitally encoded media corresponding to the manifests it distributes and necessary to respond to user requests.
[1.T] the method comprising:	

Claim Language	Contention
[1.1] reading, by at least one computer of the server system, the pre-recorded audio or video program from the computer-readable media;	WAG does not have access to direct proof that YouTube's servers read the above-described pre-recorded video programs from the above-described computer-readable media. However, a POSITA understands that in order to serve the programming recorded in the media, YouTube's servers would necessarily have to perform this claim step. WAG contends that this inference, which it does not believe YouTube will dispute, is sufficient, whether YouTube expressly concedes the point or not, to establish that the computers comprising YouTube's server system read the above-described pre-recorded video programs from the above-described computer-readable media.
[1.2.1] supplying, at the server system, media data elements representing the program,	The figure above, again, but now focusing on the right-hand side, showing the request and corresponding response,  shows one user request and corresponding server response, for one media data element in a sequence of many such elements (each similarly requested and provided) comprising the requested program stream (e.g., the entire requested video, which can go on for many minutes or in some cases hours), from beginning to end. The media data element requested in the above figure is one of a plurality of media data elements representing the requested program.

Claim Language	Contention																																																																																																																																																																																
	WAG does not have access to direct proof that YouTube supplies, at its server system, media data elements corresponding to these requests. However, a POSITA understands that in order to serve the media data elements comprising the requested program, and have them received, as indicated by the “200 OK” “success” HTTP response code, they each necessarily must be supplied at the server system. The manifest noted above also shows that the server supplies the media data elements representing the program. WAG contends that this inference, based on an observed series of user requests for such elements, and corresponding responses, proximate to each other in time, which it does not believe YouTube will dispute, is sufficient, whether YouTube expressly concedes the point or not, to establish that the computers comprising YouTube’s server system perform the step, under YouTube’s direction and control, of supplying each requested element to the server in order for WAG to have been able to observe and record the corresponding responses.																																																																																																																																																																																
[1.2.2] each media data element comprising a digitally encoded portion of the program	In the above example, the requested element comprises 2,034,266 bytes (1.94MB in binary), representing a portion (one of a large number requested over the course of receiving the program) of the requested program. The metadata shown (200 OK) reflects that the element received is what was requested, corresponding to the request parameters (including range: 9075548-11109813 and rn: 102). This represents one portion of the program, as reflected by the following summary timeline for the series of requests that included the above individual request (the recorded section representing 21 successive audio and video requests for media having 231.3 seconds duration). The following represents requests for the period recorded in the portion of the session shown in this data capture. The above individual request/response (for 1.94MB) corresponds to the first line of this summary: <table><thead><tr><th>#</th><th>Resource</th><th>Host</th><th>Mime Type</th><th>Code</th><th>Header</th><th>Body</th><th>Time</th></tr></thead><tbody><tr><td>1</td><td>videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com</td><td>videoplayback</td><td>video/mp4</td><td>200 0 0</td><td>1.94 MB</td><td>14.6s</td></tr><tr><td>2</td><td>videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com</td><td>videoplayback</td><td>audio/webm</td><td>200 0 0</td><td>154.89 KB</td><td>1.0s</td></tr><tr><td>3</td><td>videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com</td><td>videoplayback</td><td>audio/webm</td><td>200 0 0</td><td>480.25 KB</td><td>3.1s</td></tr><tr><td>4</td><td>videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com</td><td>videoplayback</td><td>video/mp4</td><td>200 0 0</td><td>1.61 MB</td><td>10.3s</td></tr><tr><td>5</td><td>videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com</td><td>videoplayback</td><td>audio/webm</td><td>200 0 0</td><td>480.25 KB</td><td>3.1s</td></tr><tr><td>6</td><td>videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com</td><td>videoplayback</td><td>video/mp4</td><td>200 0 0</td><td>1.93 MB</td><td>12.2s</td></tr><tr><td>7</td><td>videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com</td><td>videoplayback</td><td>audio/webm</td><td>200 0 0</td><td>480.25 KB</td><td>3.1s</td></tr><tr><td>8</td><td>videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com</td><td>videoplayback</td><td>video/mp4</td><td>200 0 0</td><td>1.96 MB</td><td>11.9s</td></tr><tr><td>9</td><td>videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com</td><td>videoplayback</td><td>video/mp4</td><td>200 0 0</td><td>1.93 MB</td><td>12.4s</td></tr><tr><td>10</td><td>videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com</td><td>videoplayback</td><td>audio/webm</td><td>200 0 0</td><td>470.85 KB</td><td>3.2s</td></tr><tr><td>11</td><td>videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com</td><td>videoplayback</td><td>video/mp4</td><td>200 0 0</td><td>1.64 MB</td><td>9.8s</td></tr><tr><td>12</td><td>videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com</td><td>videoplayback</td><td>audio/webm</td><td>200 0 0</td><td>480.70 KB</td><td>2.9s</td></tr><tr><td>13</td><td>videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com</td><td>videoplayback</td><td>video/mp4</td><td>200 0 0</td><td>1.62 MB</td><td>9.6s</td></tr><tr><td>14</td><td>videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com</td><td>videoplayback</td><td>audio/webm</td><td>200 0 0</td><td>487.58 KB</td><td>2.9s</td></tr><tr><td>15</td><td>videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com</td><td>videoplayback</td><td>video/mp4</td><td>200 0 0</td><td>1.64 MB</td><td>9.8s</td></tr><tr><td>16</td><td>videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com</td><td>videoplayback</td><td>video/mp4</td><td>200 0 0</td><td>1.63 MB</td><td>10.8s</td></tr><tr><td>17</td><td>videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com</td><td>videoplayback</td><td>audio/webm</td><td>200 0 0</td><td>507.89 KB</td><td>3.1s</td></tr><tr><td>18</td><td>videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com</td><td>videoplayback</td><td>video/mp4</td><td>200 0 0</td><td>1.62 MB</td><td>9.6s</td></tr><tr><td>19</td><td>videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com</td><td>videoplayback</td><td>audio/webm</td><td>200 0 0</td><td>458.49 KB</td><td>2.8s</td></tr><tr><td>20</td><td>videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com</td><td>videoplayback</td><td>video/mp4</td><td>200 0 0</td><td>1.74 MB</td><td>10.4s</td></tr><tr><td>21</td><td>videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com</td><td>videoplayback</td><td>audio/webm</td><td>200 0 0</td><td>480.32 KB</td><td>2.7s</td></tr><tr><td>Total</td><td></td><td></td><td></td><td>0 0</td><td>23.84 MB</td><td>140.4s</td></tr><tr><td>Grand Total</td><td></td><td></td><td></td><td></td><td>23.84 MB</td><td>231.3s</td></tr><tr><td>Duration</td><td></td><td></td><td></td><td></td><td></td><td></td></tr></tbody></table>	#	Resource	Host	Mime Type	Code	Header	Body	Time	1	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	video/mp4	200 0 0	1.94 MB	14.6s	2	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	audio/webm	200 0 0	154.89 KB	1.0s	3	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	audio/webm	200 0 0	480.25 KB	3.1s	4	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	video/mp4	200 0 0	1.61 MB	10.3s	5	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	audio/webm	200 0 0	480.25 KB	3.1s	6	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	video/mp4	200 0 0	1.93 MB	12.2s	7	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	audio/webm	200 0 0	480.25 KB	3.1s	8	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	video/mp4	200 0 0	1.96 MB	11.9s	9	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	video/mp4	200 0 0	1.93 MB	12.4s	10	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	audio/webm	200 0 0	470.85 KB	3.2s	11	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	video/mp4	200 0 0	1.64 MB	9.8s	12	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	audio/webm	200 0 0	480.70 KB	2.9s	13	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	video/mp4	200 0 0	1.62 MB	9.6s	14	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	audio/webm	200 0 0	487.58 KB	2.9s	15	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	video/mp4	200 0 0	1.64 MB	9.8s	16	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	video/mp4	200 0 0	1.63 MB	10.8s	17	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	audio/webm	200 0 0	507.89 KB	3.1s	18	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	video/mp4	200 0 0	1.62 MB	9.6s	19	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	audio/webm	200 0 0	458.49 KB	2.8s	20	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	video/mp4	200 0 0	1.74 MB	10.4s	21	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	audio/webm	200 0 0	480.32 KB	2.7s	Total				0 0	23.84 MB	140.4s	Grand Total					23.84 MB	231.3s	Duration						
#	Resource	Host	Mime Type	Code	Header	Body	Time																																																																																																																																																																										
1	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	video/mp4	200 0 0	1.94 MB	14.6s																																																																																																																																																																											
2	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	audio/webm	200 0 0	154.89 KB	1.0s																																																																																																																																																																											
3	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	audio/webm	200 0 0	480.25 KB	3.1s																																																																																																																																																																											
4	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	video/mp4	200 0 0	1.61 MB	10.3s																																																																																																																																																																											
5	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	audio/webm	200 0 0	480.25 KB	3.1s																																																																																																																																																																											
6	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	video/mp4	200 0 0	1.93 MB	12.2s																																																																																																																																																																											
7	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	audio/webm	200 0 0	480.25 KB	3.1s																																																																																																																																																																											
8	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	video/mp4	200 0 0	1.96 MB	11.9s																																																																																																																																																																											
9	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	video/mp4	200 0 0	1.93 MB	12.4s																																																																																																																																																																											
10	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	audio/webm	200 0 0	470.85 KB	3.2s																																																																																																																																																																											
11	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	video/mp4	200 0 0	1.64 MB	9.8s																																																																																																																																																																											
12	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	audio/webm	200 0 0	480.70 KB	2.9s																																																																																																																																																																											
13	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	video/mp4	200 0 0	1.62 MB	9.6s																																																																																																																																																																											
14	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	audio/webm	200 0 0	487.58 KB	2.9s																																																																																																																																																																											
15	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	video/mp4	200 0 0	1.64 MB	9.8s																																																																																																																																																																											
16	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	video/mp4	200 0 0	1.63 MB	10.8s																																																																																																																																																																											
17	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	audio/webm	200 0 0	507.89 KB	3.1s																																																																																																																																																																											
18	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	video/mp4	200 0 0	1.62 MB	9.6s																																																																																																																																																																											
19	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	audio/webm	200 0 0	458.49 KB	2.8s																																																																																																																																																																											
20	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	video/mp4	200 0 0	1.74 MB	10.4s																																																																																																																																																																											
21	videoplayback?expire=19287022&ie=In7Yx0Hb4q-8P4_2a4AM8p=107.77.205.94&id=0-AH9yh6ZETKAiTGtGLA...r1--sn-e087znk-googlevideo.com	videoplayback	audio/webm	200 0 0	480.32 KB	2.7s																																																																																																																																																																											
Total				0 0	23.84 MB	140.4s																																																																																																																																																																											
Grand Total					23.84 MB	231.3s																																																																																																																																																																											
Duration																																																																																																																																																																																	

Claim Language	Contention
[1.2.3] and having a playback rate;	The requested media data element is digitally encoded at a bitrate range and framerate such that a normal rendition occurs when played back at that framerate, <i>i.e.</i> , when each is played back in the corresponding time period set out in the above listing.
[1.3.1] serially identifying the media data elements,	The elements in this example are serially identified by parameters including “range” and “rn” as specified in the request headers and query string. Such identification takes place at least when the data element is stored on a storage device accessible to the server, when a manifest is created, and when the server responds to a request. The “range” and “rn” values, both individually and as an ordered pair, literally provide serial identification of the media data elements. Nonetheless, this limitation may also be found under the doctrine of equivalents. In particular, the function of serially identifying the media data elements is to uniquely identify the time-sequenced media data elements from each other, which is exactly what the “range” and “rn” values do, both individually and as an ordered pair. The way in which the media data elements are uniquely identified is to provide each with a unique identifier that can be used in requests from the client to the server; the unique “range” and “rn” values associated with each media data element perform this function, both individually and as an ordered pair. The result of the serial identification is that the client and server agree upon identifications that allow the client to uniquely request a media data element in the proper time sequence, which is what occurs when the client uses the “range” and/or “rn” values to specify a desired media data element.
[1.3.2] said serial identification indicating a time sequence of the media data elements;	The time-sequenced character of the successive elements in a transmission is reflected in the summary timeline above, again:  The rn numbers generally increment by 1 for each request. (The numbering for videos on the YouTube Web Site counts both the video and interleaved audio segments, and occasionally a number can be missing.) In this sequence, the “range” identifiers increment

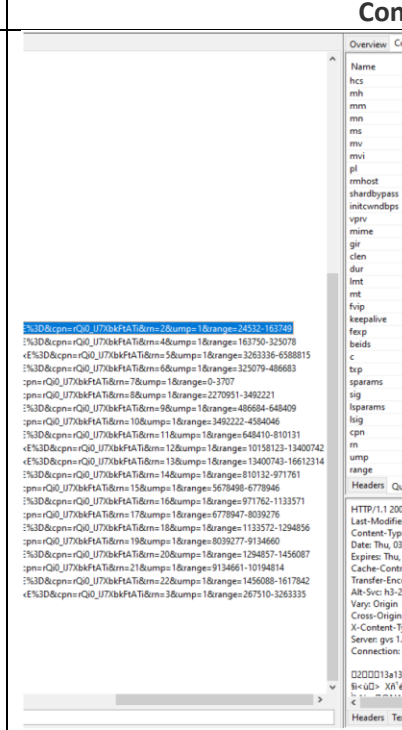
Claim Language	Contention
	for each successive segment. After initialization, the rn numbers likewise increment by 1 for each segment, in this case for segment 104 and each following segment. An example later in this sequence is rn=110:  The same can also be directly observed (without the intermediate proxy) on a full-featured browser used to access the YouTube Streaming Services, by invoking its Developer Tools (typically, Ctrl-Shift-I). The above pattern, showing a continuous time sequence of elements comprising the requested stream, corresponding to the serial identification of the elements, is representative of all request-response sequences observed with the YouTube Streaming Services, regardless of the time of observation during the period from patent issuance through March 2021, regardless of the CDN being utilized, and regardless of the user media playing platform. In many cases (as shown here) the audio is part of the numbering, and in others (e.g., live, in a separate claim chart) the rn numbers are just for the video, and likewise, for the audio, in the case of music streaming. In all cases, the range numbers also progress serially. The following are corresponding request-response timelines for other platforms streaming from YouTube Streaming Services: Google Music (Rod Stewart song Maggie May, requests/responses captured in Chrome Developer Console on a desktop computer:

Claim Language	Contention
	 First segment (above), range: 0-66403, rn: 1 Second segment, range: 66404, rn: 2 (both next in sequence); the following videoplayback elements follow the same pattern. iOS:

Claim Language

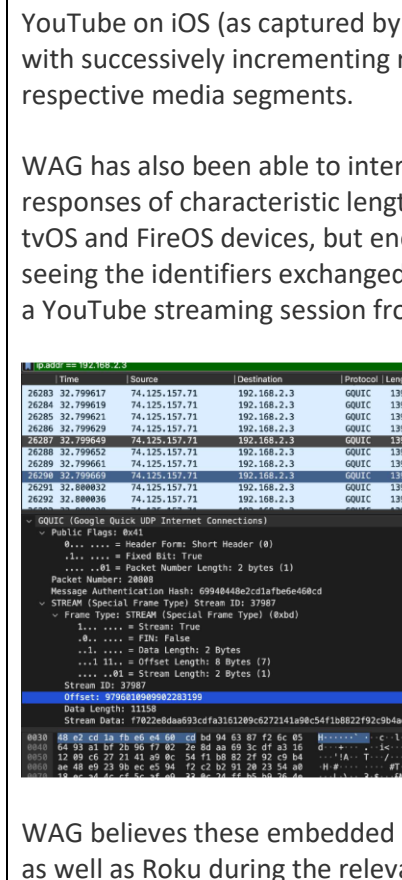
Contention





YouTube on iOS (as captured by a proxy) shows the same patterns, with successively incrementing range and rn identifiers for the respective media segments.

WAG has also been able to intercept successive requests and responses of characteristic length for streaming YouTube media to tvOS and FireOS devices, but encryption has prevented WAG from seeing the identifiers exchanged. For example, a packet capture for a YouTube streaming session from tvOS showed the following:



WAG believes these embedded platforms (e.g., tvOS, FireOS, etc., as well as Roku during the relevant period) operated with YouTube in substantially the same way as observed for desktop and iOS

Claim Language	Contention
	devices. WAG will include in its discovery requests inquiries directed to the additional platforms whose communications have been obscured by encryption. (The YouTube Live transmissions, covered in a separate claim chart) have “sq” numbers in the requests (in addition to rn), which are also sequential.) Additionally, the list of media data elements in the manifest noted above, each uniquely identified by its respective range number, also provides serial identification indicating the time sequences of the respective media data elements. Hence, not only is time sequencing of the media data elements indicated by progressive increasing of the “range” and/or “rn” values, but also by the relative positioning of the identified media data element within the manifest, with media data elements appearing higher (or first) in the manifest being earlier in time.
[1.4] storing the media data elements in a data structure under the control of the server system;	As noted above, the elements in streaming media requests to YouTube’s Streaming Services are serially identified by serial identifiers including “range” and “rn” as specified in the request and response. WAG cannot directly see how YouTube stores the media data elements comprising its recorded programs in data structures on (or under the control of) its servers. However, WAG believes that, since the requests are successful, that corresponding elements (which conform to the requested ranges and rn parameters) are stored on the server. Moreover, as discussed above, the captured manifest data also reflects that the media data elements are stored on the server in a folder/file data structure (or in an equivalent layout in a database or other data structure), wherein the folder is named “range” and the respective files have the corresponding byte range (“[number]-[number]”) as a filename.
[1.5.1] receiving requests at the server system via one or more data connections over the Internet,	The diagrams above (both the requests and responses, and the associated timeline) show that the intermediate proxy intercepted requests being made from a user system over Internet data connections, to YouTube servers, and time-proximate response thereto, reflecting the same serial identification as the corresponding request. WAG contends it is therefore reasonable to infer that the YouTube server received the request.
[1.5.2] for one or more of the media data elements stored in the data structure,	See examples in 1.3.2. The shown GET requests for the element identified by serial identifiers, including “range” and “rn” (and in some cases by both rn and sq numbers), from a specified resource (in the URL) reflects receiving a request for one or more media data

Claim Language	Contention
	elements stored in the data structure. The MPEG-DASH specification also provides for the client to request one or more media data elements stored in a data structure on the server, and the observed YouTube manifests (see example above) confirm compliance with that aspect of the specification.
[1.5.3] each received request specifying one or more serial identifiers of the requested one or more media data elements,	See 1.5.2.
[1.5.4] each received request originating from a requesting user system of the one or more user systems; and	The intercepting proxy (or alternately the browser's Developer Tools console) verifies that the request came a requesting user system.
[1.6.1] responsive to the requests, sending, by the server system, the one or more media data elements having the one or more specified serial identifiers,	Reflected in all of the responses intercepted by the intermediate proxy, as shown in the various diagrams herein, and directly reflected in Developer Tools windows on the browsers that support such tools. The result code 200 OK shown in the captures confirm success in this regard.
[1.6.2] to the requesting user systems corresponding to the requests;	The responses corresponding to the requests go the requesting user system; see 1.6.1. The response can be viewed (as data) in the Developer Console, as well as rendered on the user device display and audio output.
[1.W] wherein	
[1.W.1] the data connection between the server system and each requesting user system has a data rate more rapid than the playback rate of the one or more media data elements sent via that connection;	As seen with respect to 1.3.2, requests are made every few seconds (reflecting that the elements corresponding to the element serial ID take the corresponding number of seconds to play back), whereas element is generally transmitted in well under a second. It is reasonable to infer, therefore, that the data connection had a data rate in excess of the playback rate.
[1.W.2] each sending is at a transmission rate as fast as the data connection between the server system and each requesting user system allow;	WAG does not have access to YouTube's servers, but states that is not aware of, and sees no evidence of, any process on YouTube's servers, or any routine in the client-side code that YouTube supplies for the user devices, that in any way throttles the data connection. WAG contends that it is therefore reasonable to infer that as a result of the observed time sequence of requests and responses, and the inferred playback duration of each of the elements, that

Claim Language	Contention
	each sending is at a transmission rate as fast as the data connection between the server system and each requesting user system allow. Although this limitation is literally met, it can also be found under the doctrine of equivalents. The function of this sending step is to ensure that the media data elements are received by the user system fast enough to prevent interruptions to the playback on the user device; YouTube's servers, or the CDN servers under YouTube's control, send the media data elements to the client devices fast enough to prevent interruptions to playback, as evidenced by the uninterrupted playback on the client end. The way in which this sending is performed is by providing the media data elements to the data connection between the server system and each requesting user system that has a data rate that is more rapid than the playback rate of the media data elements sent via that connection, which, as indicated above in 1.W.1, YouTube's servers, or the CDN's under YouTube's control, also do. The result is uninterrupted playing of the time sequenced media data elements on the client device, which, as indicated, is observed on the client device.
[1.W.3] the one or more media data element sent are selected without depending on the server system maintaining a record of the last media data element sent to the requesting user systems;	As reflected in the data shown herein, the observed requests for media data elements comprising a program all originate on the client side and the responses thereto are proximate in time to the corresponding request. Further, there is no observed communication from the server to the requesting user system to indicate the last media data element that the server sent to the requesting user system. Each client request specifies a fully-specified ID that is not relative to an earlier requested ID. Furthermore, the MPEG-DASH standard states "Clients may request data using the HTTP protocol from standard web servers that have no DASH-specific capabilities." ISO/IEC 23009 at ix, 11. WAG contends that is therefore reasonable to infer that the only data transmissions from the server of the program being streamed are those responsive to specific requests (by serial identifier), and that, for lack of any observed communication from the server as to the last media data element that it sent, the client's requests are made as determined on the client side, with no dependency on any record maintained by the server system of the last media data element sent, and the server's responses, each keyed to the corresponding request, likewise have no dependency on any record of the server system as to the last media data element that it sent. Although met literally, this limitation may also be found under the doctrine of equivalents. The function of the server selecting the media data elements without dependence on the server maintaining a record of the last media data element being sent is to

Claim Language	Contention
	allow the client to control the pacing and delivery of the media data elements rather than the server. As set forth above, it is the client that controls pacing and delivery of the media data elements from the YouTube servers (or the CDNs under YouTube's control). The way in which this occurs is to allow the client to issue requests for media data elements uniquely identified by serial identifiers and for the server to respond to these requests with the corresponding media data elements. As set forth above, this is what YouTube's servers (or the CDN's under YouTube's control) do. The result is that the client primarily controls what media data elements are sent and received, and when. As indicated above, this is what happens with YouTube's streaming services.
[1.W.4] all of the media data elements that are sent by the server system to the one or more user systems are sent in response to the requests; and	WAG has observed no transmission of streaming media data from any of the YouTube Streaming Services other than those in direct response to user-side requests by serial ID, as discussed throughout the foregoing. The streaming media received, from beginning to end of the program, is 100% via pull, with each media data element being sent in response to a request therefor.
[1.W.5] all of the media data elements that are sent by the server system to the requesting user systems are sent from the data structure under the control of the server system as the media data elements were first stored therein.	See 1.4. WAG contends that the media data elements are stored on YouTube's servers as reflected in the client requests, as indicated in a manifest, and that the media data elements so structured are served to the requesting user systems as first stored.
2. The method of claim 1 wherein the serial identifiers are sequential.	As reflected in the data reproduced herein, YouTube's serial identifiers are generally also sequential. There are numerous examples of sequential runs of identifiers.
3. The method of claim 1, wherein the sending is via a reliable transmission protocol.	The combination of HTTP and TLS as reflected in the data herein indicates the use of a reliable transmission protocol.
4. The method of claim 3, wherein the reliable transmission protocol is TCP.	Some of the observed sessions used TCP as the reliable transmission protocol (others used Google QUIC, an alternate reliable transmission protocol).

Claim Language	Contention
5. A server system for distributing a pre-recorded audio or video program over the Internet to one or more user systems, the server system comprising:	See 1.P.1; the claim here is for the server that performs the method of claim 1.
at least one computer having a connection to the Internet;	YouTube's server is reachable over the Internet and it is therefore reasonable to infer that it comprises at least one computer having a connection to the Internet.
a machine-readable, executable routine containing instructions to cause one of the at least one computers to read the pre-recorded audio or video program from computer-readable media,	See 1.1.
to supply, at the server system, media data elements representing the program, each media data element comprising a digitally encoded portion of the program and having a playback rate,	See 1.2.1-1.2.3.
to serially identify the media data elements, said serial identification indicating a time sequence of the media data elements, and	See 1.3.1-1.3.2.
to store the media data elements in a data structure under the control of the server system;	See 1.4.
a machine-readable, executable routine containing instructions to cause one of the at least one computers to receive requests at the server system via one or more data connections over the Internet, for one or more of the media data elements stored in the data structure, each received request specifying one or more	See 1.5.1-1.5.4.

Claim Language	Contention
serial identifiers of the one or more media data elements, each received request originating from a requesting user system of the one or more user systems; and	
a machine-readable, executable routine containing instructions to cause one of the at least one computers to send, responsive to the requests, the one or more media data elements having the one or more specified serial identifiers, to the requesting user systems corresponding to the requests; wherein	See 1.6.1-1.6.2.
the data connection between the server system and each requesting user system has a data rate more rapid than the playback rate of the one or more media data elements sent via that connection;	See 1.W.1.
each sending is at a transmission rate as fast as the data connection between the server system and each requesting user system allow;	See 1.W.2.
the one or more media data elements sent are selected without depending on the server system maintaining a record of the last media data element sent to the requesting user systems;	See 1.W.3.
all of the media data elements that are sent by the server to the one or more user systems are sent in response to the requests; and	See 1.W.4.
all of the media data elements that are sent by the server system to the requesting user systems are sent from the data structure under the control of	See 1.W.5

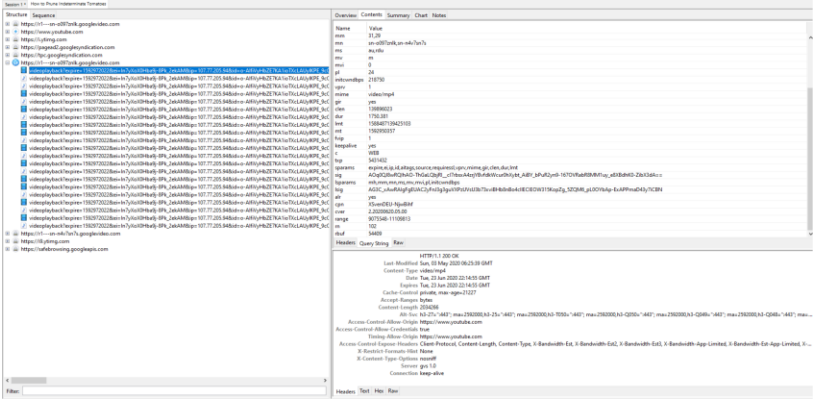
Claim Language	Contention
the server system as the media data elements were first stored therein.	
6. The server system of claim 5 wherein the serial identifiers are sequential.	See 5 and 2.
7. The server system of claim 5, wherein the sending is via a reliable transmission protocol.	See 5 and 3.
8. The server system of claim 5, wherein the reliable transmission protocol is TCP.	See 5 and 4.
[9.P] A computer program product for distributing over the Internet from a server system comprising at least one computer to one or more user systems, a pre-recorded audio or video program stored in digitally encoded form on computer-readable media, the computer program product comprising a non-transitory computer readable storage medium having program instructions embodied therewith, the program instructions comprising:	See 1.P.1; the claim here is for a computer program/non-transitory media embodying program instructions that implement the method of claim 1.
[9.1] instructions executable to cause one of the at least one computers to read the pre-recorded audio or video program from the computer-readable media,	See 1.1.
[9.1.1] to supply, at the server system, media data elements representing the program, each media data element comprising a digitally encoded portion of the program and having a playback rate,	See 1.2.1-1.2.3.

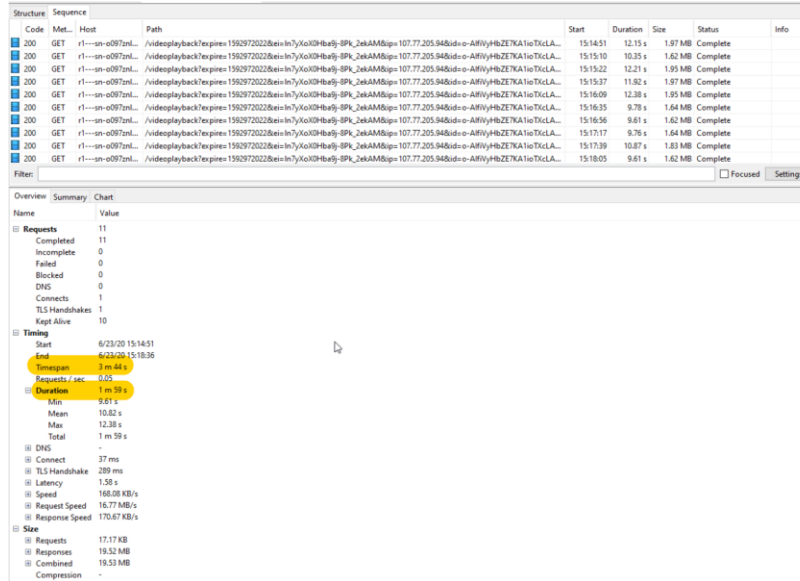
Claim Language	Contention
[9.1.2] to serially identify the media data elements, said serial identification indicating a time sequence of the media data elements, and	See 1.3.1-1.3.2.
[9.1.3] to store the media data elements in a data structure under the control of the server system;	See 1.4.
[9.2] instructions executable to cause one of the at least one computers to receive requests at the server system via one or more data connections over the Internet, for one or more of the media data elements stored in the data structure, each received request specifying one or more serial identifiers of the one or more media data elements, each received request originating from a requesting user system of the one or more user systems; and	See 1.5.1-1.5.4.
[9.3] instructions executable to cause one of the at least one computers to send, responsive to the requests, the one or more media data elements having the one or more specified serial identifiers to the requesting user systems corresponding to the requests; wherein	See 1.6.1-1.6.2.
[9.W.1] the data connection between the server system and each requesting user systems has a data rate more rapid than the playback rate of the one or more media data elements sent via that connection;	See 1.W.1.
[9.W.2] each sending is at a transmission rate as fast as the data connection between the	See 1.W.2.

Claim Language	Contention
server system and each requesting user system allows;	
[9.W.3] the one or more media data elements sent are selected without depending on the server system maintaining a record of the last media data element sent to the requesting user systems;	See 1.W.3.
[9.W.4] all of the media data elements that are sent by the server to the one or more user systems are sent in response to the requests; and	See 1.W.4.
[9.W.5] all of the media data elements that are sent by the server system to the requesting user systems are sent from the data structure under the control of the server system as the media data elements were first stored therein.	See 1.W.5.
10. The computer program product of claim 9 wherein the serial identifiers are sequential.	See 9 and 2.
11. The computer program product of claim 9, wherein the sending is via a reliable transmission protocol.	See 9 and 3.
12. The computer program product of claim 11, wherein the reliable transmission protocol is TCP.	See 9 and 4.

Claim Language	Contention
Overview	WAG has asserted three patents against YouTube, which concern streaming media over the Internet, conducted between a user (client) system, which requests and plays the media, and a server to which the client system connects over the Internet, which provides the requested media. WAG's '824 and '636 patents concern methods, apparatus, and computer program products on the server side of this interaction, for prerecorded and live media, respectively (which differ, in that the latter concerns media created contemporaneously with its streaming). WAG's '594 patent (the subject of the within contentions) concerns methods, apparatus, and computer program products on the client side of the above-described interaction, which commonly apply whether the server is providing prerecorded or live media. Because the subject matter of these patents overlap, WAG's infringement contentions with regard to the '824 and '636 patents are each incorporated herein by reference. YouTube LLC and/or affiliated entities such as Google LLC ("YouTube") directs and controls the activity of streaming digitally encoded media it provides over the Internet through the site youtube.com (also accessible as www.youtube.com) (the "YouTube Web Site") from servers owned and/or the operations of which are controlled by YouTube, including servers in the U.S. and in the Western District of Texas ("YouTube's Servers"). YouTube also distributes pre-recorded video programs from YouTube's servers, over the Internet, to large numbers of users in the U.S. and elsewhere. This is done via a pull protocol defined at the application layer and generally conforming with the MPEG-DASH specification, ISO/IEC 23009 (citations are to the fourth edition, Dec. 2009 (cited herein as "ISO/IEC 23009"), though WAG does not believe that any versions applicable during the period of the alleged infringement had any material differences). Google is a Charter Member of the DASH Industry Forum. Google's implementation of MPEG-DASH has been the subject of some academic work, including the paper Mondal, et al., "Candid with YouTube: Adaptive Streaming Behavior and Implications on Data Consumption," Proceedings of the Workshop on Network and Operating Systems Support for Digital Audio and Video: 19-24 Taipei, Taiwan: Association for Computing Machinery, doi: 10.1145/3083165.3083177 (2017), source: https://dl.acm.org/citation.cfm?doid=3083165.3083177 isbn: 978-1-4503-5003-7. YouTube's web pages also contain reference to a manifest for HTTP Live Streaming ("HLS"), in addition to providing a link to an MPEG-DASH manifest. HLS is a streaming media delivery specification,

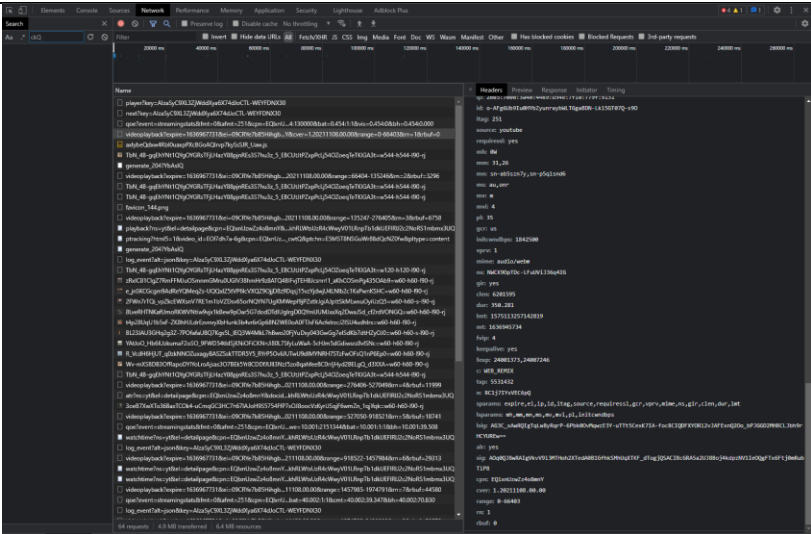
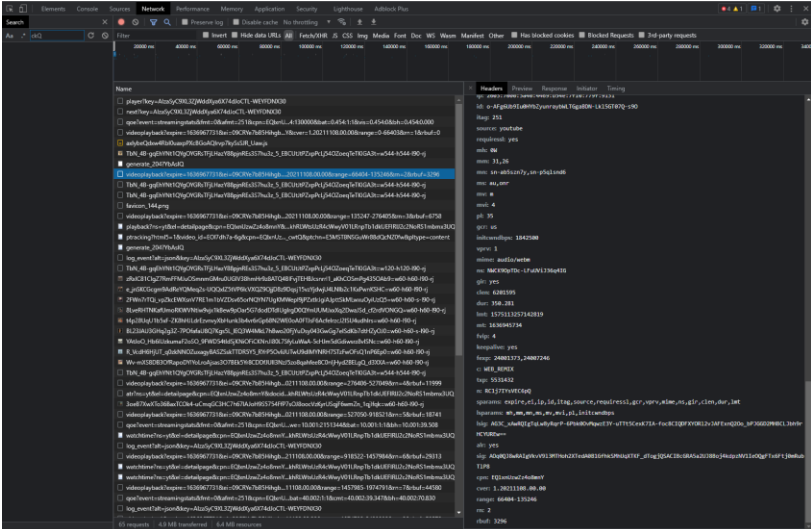
Claim Language	Contention
	compliance with which infringes WAG's patents asserted herein on substantially the same facts shown herein with respect to YouTube's use of MPEG-DASH. The client-side JavaScript files that YouTube distributes in support of its streaming media (<i>e.g.</i>, <i>base.js</i>) also provide functions for HLS. WAG contends that YouTube has supported streaming media delivery during the relevant period via HLS as at least a fallback where the user device identifies itself as a device that normally will not support MPEG-DASH or prefers HLS. Under the MPEG-DASH standard (ISO/IEC 23009 at ix), control of streaming media delivery lies primarily with the client, and YouTube's accused streaming reflects the same characteristic, sometimes referred to as "Pull." The server is responsible for dividing, storing, and identifying streaming content data elements to be responsive to requests it will receive from clients, but the timing and mechanics of delivering the requested elements is determined by processes that run on the client. To make such a "pull" delivery scheme work, YouTube must ensure that software that is compatible with the manner in which the content is stored on YouTube's servers is operable on the client side. YouTube does this in two different ways. First, for user-agents (clients) that can execute JavaScript or a comparable language, YouTube provides from its servers JavaScript files that implement the desired content-requesting functionality. This can be done as part of the streaming session. For other user-agents, such as embedded user-agents (<i>e.g.</i>, various Roku devices, tvOS, FireOS, iOS, Android, etc.), YouTube provides a comparable native application for the device, for example, through the device provider's app store. These contentions relate to every streaming video client-side program provided through the YouTube Site or by YouTube as integrated into devices it makes or sells, or provided by YouTube to app stores for the respective devices, for the period from the 2017 issuance date of the patent addressed in these contentions, through the common date of expiration of the patent (March 28, 2021). The contentions apply to the software and devices streaming either pre-recorded or live video. Also included in these contentions is streaming audio distribution from the YouTube TV subscription service (available for distribution through tv.youtube.com), and music.youtube.com (formerly Google Play Music). The foregoing services are addressed individually, but are also referred to, for purposes of collective reference, as the "YouTube Streaming Services."

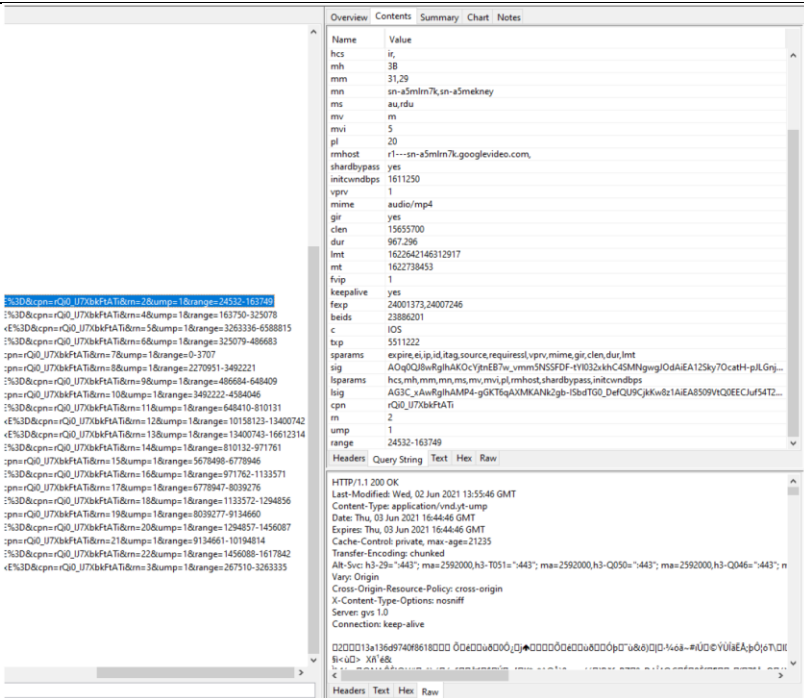
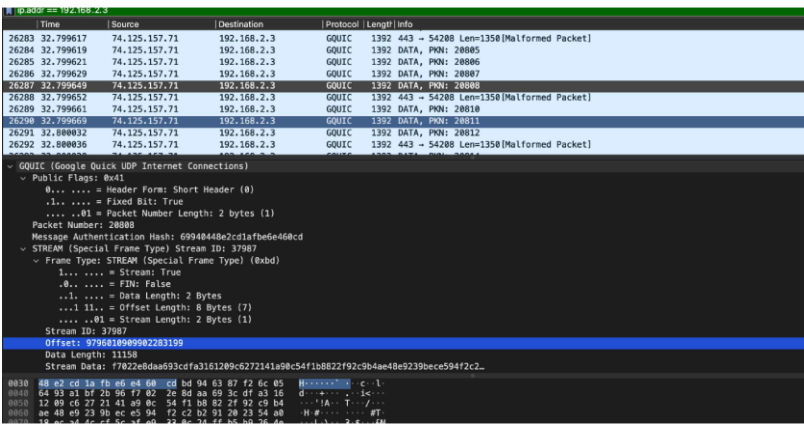
Claim Language	Contention
	The web debugging proxy described in connection with the '824 patent is also relevant to the '594 patent, in that, for purposes of investigation, the client device and its software relevant to the '594 patent sits behind the proxy. In scenarios involving a desktop browser client (e.g., Chrome) the operations of the client may be monitored directly through the browser's Developer Console functionality (or the like). As seen with respect to the '824 patent (charted separately), a streaming session involves a set of requests/responses illustrated by the following actual example of a client-server request/response sequence for a program on YouTube as captured in June 2020 by a web debugging proxy:  The entirety of each requested video program (from beginning to end) is delivered as a series of time-sequenced media data elements, where each such element has been given one or more serial identifiers (e.g., range=9075548-11109813, rn=102), and requested by serial identifier, as reflected by the sample sequence above, reflecting 11 such successive requests, interleaved with similarly sequenced requests for the audio portion of the program. The client-side software in these examples is responsible for requesting each such element, by its serial identifier, when and as needed by the client, in order to sustain continuous and uninterrupted playback. Because the streaming media flow as to each element in the stream is thus dependent on specific client-side requests for the respective elements comprising the stream, the overall process is sometimes referred to as a “pull.” Furthermore, in order for the client to successfully keep up with the stream while making such pull requests, it should not take longer to request and receive the individual segments than the time it takes to play the segments back at a normal rendition. Repeated

Claim Language	Contention																																																																
	observations confirm that transmissions from YouTube Servers consistently meet these requirements. For example, in the below sequence, requesting and receiving video data having 3m 44 sec of playback time only uses 1m 59sec. for the transfer, meaning that the data is transferred in about half the time it takes for the client system to play it back:  The screenshot shows a sequence of 11 GET requests to YouTube servers. The summary below the sequence provides the following data: <table border="1"> <thead> <tr> <th>Category</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>Requests</td> <td>11</td> </tr> <tr> <td>Completed</td> <td>11</td> </tr> <tr> <td>Incomplete</td> <td>0</td> </tr> <tr> <td>Failed</td> <td>0</td> </tr> <tr> <td>Blocked</td> <td>0</td> </tr> <tr> <td>DNS</td> <td>0</td> </tr> <tr> <td>Connects</td> <td>1</td> </tr> <tr> <td>TLS Handshakes</td> <td>1</td> </tr> <tr> <td>Kept Alive</td> <td>10</td> </tr> <tr> <td>Timing</td> <td></td> </tr> <tr> <td>Start</td> <td>6/23/20 15:14:51</td> </tr> <tr> <td>End</td> <td>6/23/20 15:18:36</td> </tr> <tr> <td>Transfer</td> <td>3 m 44 s</td> </tr> <tr> <td>Requests / sec</td> <td>0.05</td> </tr> <tr> <td>Duration</td> <td>1 m 59 s</td> </tr> <tr> <td>Min</td> <td>9.21 s</td> </tr> <tr> <td>Mean</td> <td>10.82 s</td> </tr> <tr> <td>Max</td> <td>12.38 s</td> </tr> <tr> <td>Total</td> <td>1 m 59 s</td> </tr> <tr> <td>DNS</td> <td>0 ms</td> </tr> <tr> <td>Connect</td> <td>37 ms</td> </tr> <tr> <td>TLS Handshake</td> <td>289 ms</td> </tr> <tr> <td>Latency</td> <td>1.58 s</td> </tr> <tr> <td>Speed</td> <td>188.08 KB/s</td> </tr> <tr> <td>Request Speed</td> <td>16.77 MB/s</td> </tr> <tr> <td>Response Speed</td> <td>170.67 KB/s</td> </tr> <tr> <td>Size</td> <td></td> </tr> <tr> <td>Requests</td> <td>17.17 KB</td> </tr> <tr> <td>Responses</td> <td>19.52 MB</td> </tr> <tr> <td>Combined</td> <td>19.53 MB</td> </tr> <tr> <td>Compression</td> <td>-</td> </tr> </tbody> </table> The MPEG-DASH standard, ISO/IEC 23009, spells out media distribution steps that align identically with each of the claim limitations herein. The version of the MPEG-DASH specification referenced herein is the Fourth edition, Dec. 2019. However, WAG believes that YouTube also complied with earlier and later versions of this standard, with no differences between such versions being material to the claims herein. The foregoing figure is representative of every platform streaming from YouTube Servers for which WAG has been able to observe the contents of HTTP(S) traffic, including Chrome, Firefox, Safari, and other desktop browsers, iOS on iPhone and iPad, and Android, as well as TVOS and FireOS. It is also representative of Google Play Music/YouTube Music, except insofar as these audio services are limited to audio media data elements (still nevertheless internally named “videoplayback”), as well as YouTube TV. On some embedded platforms (such as Roku and TVOS), due to encryption, WAG has been unable to inspect the contents of HTTPS (encrypted) requests and responses. However, the timing of the encrypted requests and responses follow the same pattern as observable on other platforms (e.g., desktop Chrome, as well as the	Category	Value	Requests	11	Completed	11	Incomplete	0	Failed	0	Blocked	0	DNS	0	Connects	1	TLS Handshakes	1	Kept Alive	10	Timing		Start	6/23/20 15:14:51	End	6/23/20 15:18:36	Transfer	3 m 44 s	Requests / sec	0.05	Duration	1 m 59 s	Min	9.21 s	Mean	10.82 s	Max	12.38 s	Total	1 m 59 s	DNS	0 ms	Connect	37 ms	TLS Handshake	289 ms	Latency	1.58 s	Speed	188.08 KB/s	Request Speed	16.77 MB/s	Response Speed	170.67 KB/s	Size		Requests	17.17 KB	Responses	19.52 MB	Combined	19.53 MB	Compression	-
Category	Value																																																																
Requests	11																																																																
Completed	11																																																																
Incomplete	0																																																																
Failed	0																																																																
Blocked	0																																																																
DNS	0																																																																
Connects	1																																																																
TLS Handshakes	1																																																																
Kept Alive	10																																																																
Timing																																																																	
Start	6/23/20 15:14:51																																																																
End	6/23/20 15:18:36																																																																
Transfer	3 m 44 s																																																																
Requests / sec	0.05																																																																
Duration	1 m 59 s																																																																
Min	9.21 s																																																																
Mean	10.82 s																																																																
Max	12.38 s																																																																
Total	1 m 59 s																																																																
DNS	0 ms																																																																
Connect	37 ms																																																																
TLS Handshake	289 ms																																																																
Latency	1.58 s																																																																
Speed	188.08 KB/s																																																																
Request Speed	16.77 MB/s																																																																
Response Speed	170.67 KB/s																																																																
Size																																																																	
Requests	17.17 KB																																																																
Responses	19.52 MB																																																																
Combined	19.53 MB																																																																
Compression	-																																																																

Claim Language	Contention
	FireOS embedded platform), and WAG believes the MPEG-DASH streaming implementation is the same in all material respects to the other observed implementations. WAG will include in its discovery requests inquiries directed to the additional platforms whose communications have been obscured by encryption. In the case of JavaScript implementations, the code is downloaded into the browser, as a file named, for example, base.js. WAG believes, based on the similarity in the timing and data transfers when viewing the same programs, that YouTube-provided embedded apps behave in the same manner as the JavaScript in all material respects. WAG will request discovery of the source code for YouTube apps to allow it to further develop these contentions with regard to the range of YouTube client software formats.
[1.P.1] A method for operating a media player to receive and play an audio or video program, from a remote media source via a data connection over the Internet,	This method is performed when operating a client to access YouTube streaming content.
[1.T] the method comprising:	
[1.1.1] sending requests from the media player to the media source via the data connection	The above capture (and others reflected below) shows a plurality of HTTP GET requests made from a media player (desktop computer running the Chrome browser) to an identified media source on YouTube, via the player's data connection to the server. This plurality of requests from the media player to the media source is also reflected in a manifest. YouTube's MPEG-DASH videos are requested by the user device, by successive segments of video and audio, pursuant to a manifest supplied to the user device by the YouTube server. Such manifests contain entries such as the following, indicating storage of the successive audio and video segments in a file system on the YouTube servers: <pre> <SegmentList> <Initialization sourceURL="range/0-258"/> <SegmentURL media="range/460-187101"/> <SegmentURL media="range/187102-376159"/> <SegmentURL media="range/376160-561636"/> <SegmentURL media="range/561637-745450"/> <SegmentURL media="range/745451-926622"/> <SegmentURL media="range/926623-1114800"/> </pre>

Claim Language	Contention
	<pre> <SegmentURL media="range/1114801-1299024"/> <SegmentURL media="range/1299025-1481544"/> <SegmentURL media="range/1481545-1664235"/> <SegmentURL media="range/1664236-1844136"/> <SegmentURL media="range/1844137-2020117"/> <SegmentURL media="range/2020118-2151945"/> </SegmentList> </pre> The above is from the manifest for the video at https://www.youtube.com/watch?v=i2lhwb_OckQ (a different video, which also reflects a series of requests/responses by range and rn as shown in the first example above). The manifest is designed, per the MPEG-DASH specification, to cause the user device to sequentially request the listed media segments, resulting in the continuous playback of the subject video. The right side of each of the above expressions represents a path to a storage location of the corresponding segment, on the YouTube server, wherein “range” represents a directory (folder), and “[number]-[number]” represents a file (or an equivalent to a folder-file arrangement, as implemented in a database or other data structure). The JavaScript or equivalent user-side program causes the media player to iterate through the provided list of segments to request, over its data connection, the segments comprising the desired stream, from the media source.
[1.1.2] for one or more serially identified media data elements representing the audio or video program,	<i>E.g.</i>, the shown request for bytes 9075548-11109813 and rn 102 represent a portion of a stream for the requested program. This pattern, wherein the client drives the streaming process by making successive requests by serial identifiers to a server for a sequence of segments that comprise an audio or video program is repeated over the entire range of on-demand and live streaming from YouTube. The following are corresponding request-response timelines for other platforms streaming from YouTube Streaming Services: Streaming from Google Music to a desktop computer (Rod Stewart song Maggie May, requests/responses captured in Chrome Developer Console on a desktop computer:

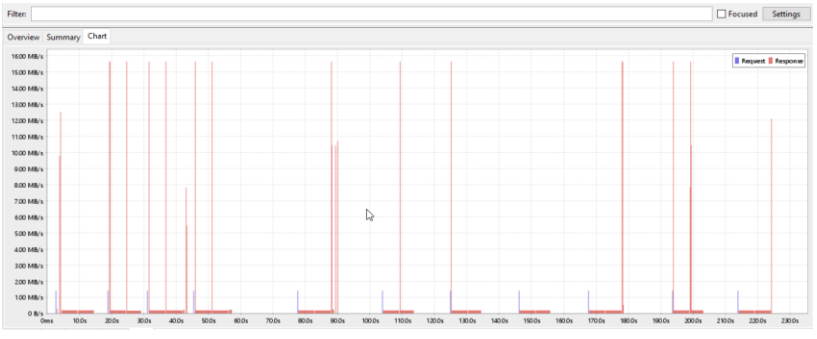
Claim Language	Contention
	 First segment (above), range: 0-66403, rn: 1  Second segment, range: 66404-1352468, rn: 2 (both next in sequence); the following video playback elements follow the same pattern. Streaming to an iOS device:

Claim Language	Contention
	 The screenshot displays a network packet capture analysis tool. The top section shows a list of media segments with their respective ranges and identifiers. The bottom section shows the details of a selected packet, including the HTTP header and the media data. Media Segments: 3:3D&cpnr=Q0_U77bKFAfATdms=3&ump=1&range=34532-163749 3:3D&cpnr=Q0_U77bKFAfATdms=4&ump=1&range=163750-325078 3:3D&cpnr=Q0_U77bKFAfATdms=5&ump=1&range=326336-658815 3:3D&cpnr=Q0_U77bKFAfATdms=6&ump=1&range=325079-486683 3:3D&cpnr=Q0_U77bKFAfATdms=7&ump=1&range=0-3707 3:3D&cpnr=Q0_U77bKFAfATdms=8&ump=1&range=227951-348222 3:3D&cpnr=Q0_U77bKFAfATdms=9&ump=1&range=486684-648409 3:3D&cpnr=Q0_U77bKFAfATdms=10&ump=1&range=348222-458406 3:3D&cpnr=Q0_U77bKFAfATdms=11&ump=1&range=648410-810131 3:3D&cpnr=Q0_U77bKFAfATdms=12&ump=1&range=10158123-13400742 3:3D&cpnr=Q0_U77bKFAfATdms=13&ump=1&range=13400743-16612314 3:3D&cpnr=Q0_U77bKFAfATdms=14&ump=1&range=810132-971761 3:3D&cpnr=Q0_U77bKFAfATdms=15&ump=1&range=967948-677896 3:3D&cpnr=Q0_U77bKFAfATdms=16&ump=1&range=971762-1133571 3:3D&cpnr=Q0_U77bKFAfATdms=17&ump=1&range=677897-8039276 3:3D&cpnr=Q0_U77bKFAfATdms=18&ump=1&range=1133572-1294856 3:3D&cpnr=Q0_U77bKFAfATdms=19&ump=1&range=8039277-9134660 3:3D&cpnr=Q0_U77bKFAfATdms=20&ump=1&range=1294857-1456087 3:3D&cpnr=Q0_U77bKFAfATdms=21&ump=1&range=913461-10194814 3:3D&cpnr=Q0_U77bKFAfATdms=22&ump=1&range=1456088-1617842 3:3D&cpnr=Q0_U77bKFAfATdms=23&ump=1&range=267510-326335 HTTP Header: HTTP/1.1 200 OK Last-Modified: Wed, 02 Jun 2021 13:55:46 GMT Content-Type: application/vnd.youtube-ump Date: Thu, 03 Jun 2021 16:44:46 GMT Expires: Thu, 03 Jun 2021 16:44:46 GMT Cache-Control: private, max-age=21235 Transfer-Encoding: chunked Alt-Svc: h3-29="443"; max=2592000,h3-1051="443"; max=2592000,h3-Q050="443"; max=2592000,h3-Q046="443"; n Vary: Origin Cross-Origin-Resource-Policy: cross-origin X-Content-Type-Options: nosniff Server: gvs 1.0 Connection: keep-alive
	YouTube on iOS (as captured by a proxy) shows the same patterns, with successively incrementing range and rn identifiers for the respective media segments. Other Embedded Devices WAG has also been able to intercept successive requests and responses of characteristic length for streaming YouTube media to tvOS and FireOS devices, but encryption has prevented WAG from seeing the identifiers exchanged. For example, a packet capture for a YouTube streaming session from tvOS showed the following:  The screenshot displays a network packet capture analysis tool. The top section shows a list of media segments with their respective ranges and identifiers. The bottom section shows the details of a selected packet, including the HTTP header and the media data. Media Segments: 3:3D&cpnr=Q0_U77bKFAfATdms=3&ump=1&range=34532-163749 3:3D&cpnr=Q0_U77bKFAfATdms=4&ump=1&range=163750-325078 3:3D&cpnr=Q0_U77bKFAfATdms=5&ump=1&range=326336-658815 3:3D&cpnr=Q0_U77bKFAfATdms=6&ump=1&range=325079-486683 3:3D&cpnr=Q0_U77bKFAfATdms=7&ump=1&range=0-3707 3:3D&cpnr=Q0_U77bKFAfATdms=8&ump=1&range=227951-348222 3:3D&cpnr=Q0_U77bKFAfATdms=9&ump=1&range=486684-648409 3:3D&cpnr=Q0_U77bKFAfATdms=10&ump=1&range=348222-458406 3:3D&cpnr=Q0_U77bKFAfATdms=11&ump=1&range=648410-810131 3:3D&cpnr=Q0_U77bKFAfATdms=12&ump=1&range=10158123-13400742 3:3D&cpnr=Q0_U77bKFAfATdms=13&ump=1&range=13400743-16612314 3:3D&cpnr=Q0_U77bKFAfATdms=14&ump=1&range=810132-971761 3:3D&cpnr=Q0_U77bKFAfATdms=15&ump=1&range=967948-677896 3:3D&cpnr=Q0_U77bKFAfATdms=16&ump=1&range=971762-1133571 3:3D&cpnr=Q0_U77bKFAfATdms=17&ump=1&range=677897-8039276 3:3D&cpnr=Q0_U77bKFAfATdms=18&ump=1&range=1133572-1294856 3:3D&cpnr=Q0_U77bKFAfATdms=19&ump=1&range=8039277-9134660 3:3D&cpnr=Q0_U77bKFAfATdms=20&ump=1&range=1294857-1456087 3:3D&cpnr=Q0_U77bKFAfATdms=21&ump=1&range=913461-10194814 3:3D&cpnr=Q0_U77bKFAfATdms=22&ump=1&range=1456088-1617842 3:3D&cpnr=Q0_U77bKFAfATdms=23&ump=1&range=267510-326335 HTTP Header: HTTP/1.1 200 OK Last-Modified: Wed, 02 Jun 2021 13:55:46 GMT Content-Type: application/vnd.youtube-ump Date: Thu, 03 Jun 2021 16:44:46 GMT Expires: Thu, 03 Jun 2021 16:44:46 GMT Cache-Control: private, max-age=21235 Transfer-Encoding: chunked Alt-Svc: h3-29="443"; max=2592000,h3-1051="443"; max=2592000,h3-Q050="443"; max=2592000,h3-Q046="443"; n Vary: Origin Cross-Origin-Resource-Policy: cross-origin X-Content-Type-Options: nosniff Server: gvs 1.0 Connection: keep-alive

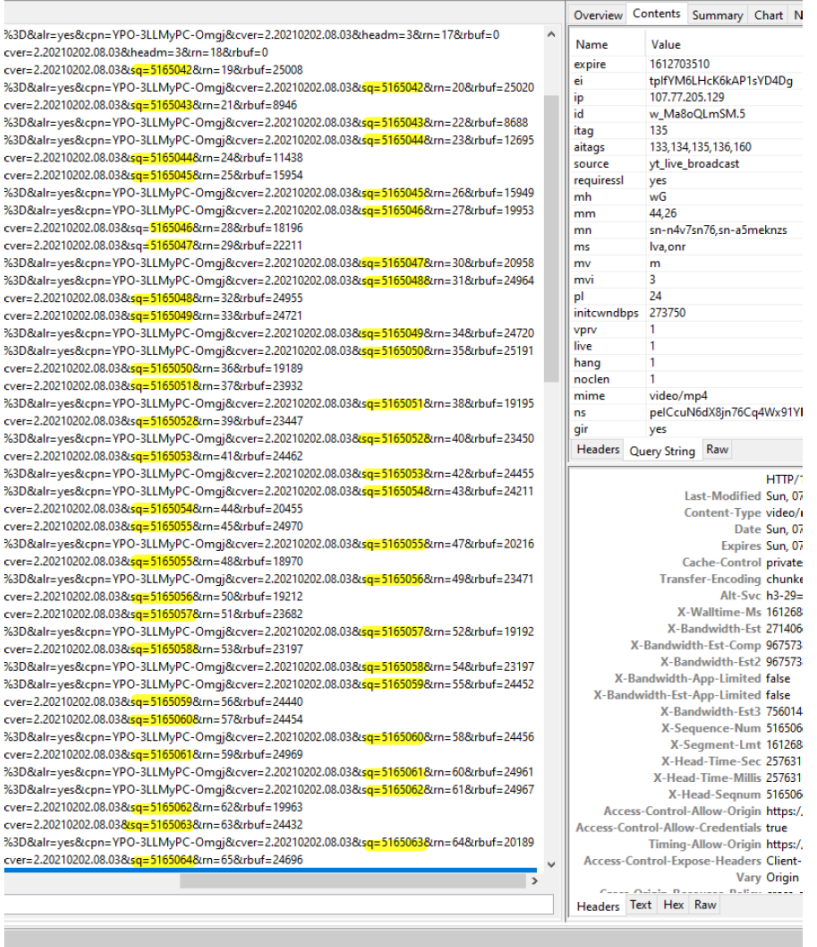
Claim Language	Contention
	WAG believes these embedded platforms (<i>e.g.</i>, tvOS, FireOS, etc., as well as Roku during the relevant period) operated with YouTube in substantially the same way as observed for desktop and iOS devices. WAG will include in its discovery requests inquiries directed to the additional platforms whose communications have been obscured by encryption. YouTube’s distribution of live programming (to the same array of consuming devices) follows the same pattern. The YouTube Live transmissions, covered in the ’636 patent claim chart, have “sq” numbers in the requests (in addition to rn), which are also sequential. YouTube’s JavaScript and Apps likewise infringe the ’594 patent for live media. For live programming, the sq numbers are used in the manifest to indicate a directory for the element files corresponding to the byte ranges. See corresponding contentions for the ’636 patent, incorporated herein by reference. The “range” and “rn” values, both individually and as an ordered pair, literally provide serial identification of the media data elements. Nonetheless, this limitation may also be found under the doctrine of equivalents. In particular, the function of serially identifying the media data elements is to uniquely identify the time-sequenced media data elements from each other, which is exactly what the “range” and “rn” values do, both individually and as an ordered pair. The way in which the media data elements are uniquely identified is to provide each with a unique identifier that can be used in requests from the client to the server; the unique “range” and “rn” values associated with each media data element perform this function, both individually and as an ordered pair. The result of the serial identification is that the client and server agree upon identifications that allow the client to uniquely request a media data element in the proper time sequence, which is what occurs when the client uses the “range” and/or “rn” values to specify a desired media data element.
[1.1.3] each media data element comprising a digitally encoded portion of the audio or video program	In the June 2020 capture referenced above (for example), the requested element comprises 2,034,266 bytes, representing a portion (one of a large number requested over the course of receiving the program) of the requested program. The captured sequence was as follows (the above capture for rn 102 (and corresponding range) corresponding to the first line below):

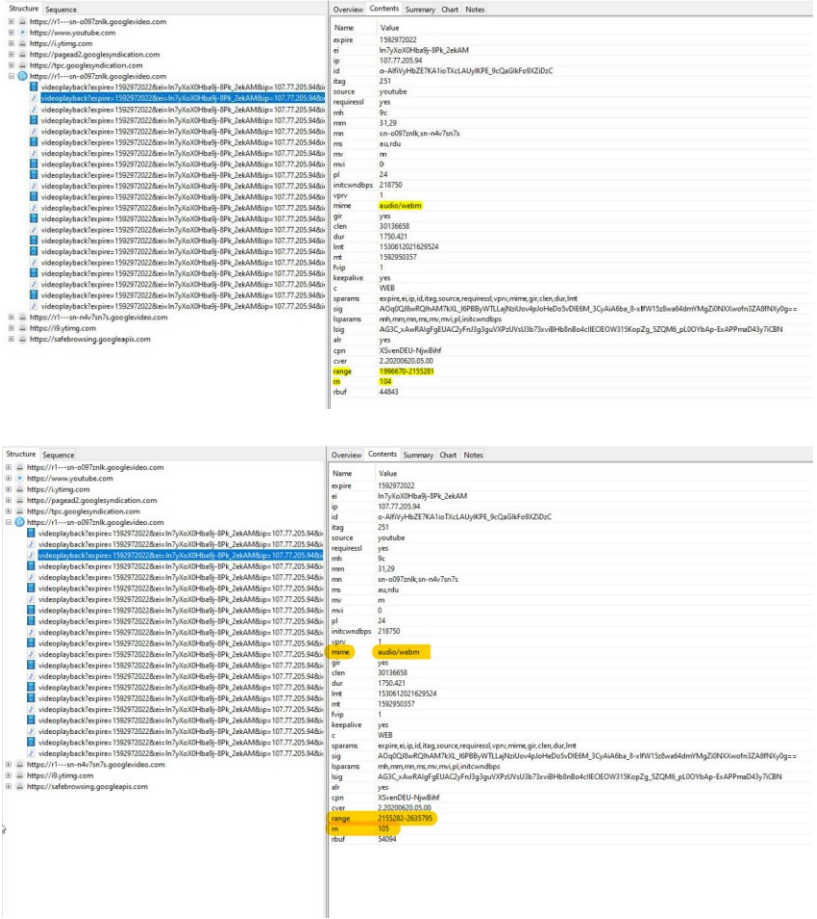
Claim Language	Contention																																																																																																																																																																																																											
	<table><tr><th>Overview</th><th>Summary</th><th>Chart</th></tr><tr><th>#</th><th>Resource</th><th>Host</th><th>Mime Type</th><th>C...</th><th>Header</th><th>Body</th><th>Time</th></tr><tr><td>1</td><td>videoplayback?expire=1592972022&ei=In7yXoX0Hba...</td><td>r1---sn-o097znlk.googlevideo...</td><td>video/mp4</td><td>200</td><td>0 B</td><td>1.94 MB</td><td>14.6s</td></tr><tr><td>2</td><td>videoplayback?expire=1592972022&ei=In7yXoX0Hba...</td><td>r1---sn-o097znlk.googlevideo...</td><td>audio/web...</td><td>200</td><td>0 B</td><td>154.89 KB</td><td>1.0s</td></tr><tr><td>3</td><td>videoplayback?expire=1592972022&ei=In7yXoX0Hba...</td><td>r1---sn-o097znlk.googlevideo...</td><td>audio/web...</td><td>200</td><td>0 B</td><td>469.25 KB</td><td>3.1s</td></tr><tr><td>4</td><td>videoplayback?expire=1592972022&ei=In7yXoX0Hba...</td><td>r1---sn-o097znlk.googlevideo...</td><td>video/mp4</td><td>200</td><td>0 B</td><td>1.61 MB</td><td>10.3s</td></tr><tr><td>5</td><td>videoplayback?expire=1592972022&ei=In7yXoX0Hba...</td><td>r1---sn-o097znlk.googlevideo...</td><td>audio/web...</td><td>200</td><td>0 B</td><td>466.26 KB</td><td>3.1s</td></tr><tr><td>6</td><td>videoplayback?expire=1592972022&ei=In7yXoX0Hba...</td><td>r1---sn-o097znlk.googlevideo...</td><td>video/mp4</td><td>200</td><td>0 B</td><td>1.95 MB</td><td>12.2s</td></tr><tr><td>7</td><td>videoplayback?expire=1592972022&ei=In7yXoX0Hba...</td><td>r1---sn-o097znlk.googlevideo...</td><td>audio/web...</td><td>200</td><td>0 B</td><td>482.05 KB</td><td>3.1s</td></tr><tr><td>8</td><td>videoplayback?expire=1592972022&ei=In7yXoX0Hba...</td><td>r1---sn-o097znlk.googlevideo...</td><td>video/mp4</td><td>200</td><td>0 B</td><td>1.96 MB</td><td>11.9s</td></tr><tr><td>9</td><td>videoplayback?expire=1592972022&ei=In7yXoX0Hba...</td><td>r1---sn-o097znlk.googlevideo...</td><td>video/mp4</td><td>200</td><td>0 B</td><td>1.95 MB</td><td>12.4s</td></tr><tr><td>10</td><td>videoplayback?expire=1592972022&ei=In7yXoX0Hba...</td><td>r1---sn-o097znlk.googlevideo...</td><td>audio/web...</td><td>200</td><td>0 B</td><td>475.85 KB</td><td>3.2s</td></tr><tr><td>11</td><td>videoplayback?expire=1592972022&ei=In7yXoX0Hba...</td><td>r1---sn-o097znlk.googlevideo...</td><td>video/mp4</td><td>200</td><td>0 B</td><td>1.64 MB</td><td>9.8s</td></tr><tr><td>12</td><td>videoplayback?expire=1592972022&ei=In7yXoX0Hba...</td><td>r1---sn-o097znlk.googlevideo...</td><td>audio/web...</td><td>200</td><td>0 B</td><td>480.70 KB</td><td>2.9s</td></tr><tr><td>13</td><td>videoplayback?expire=1592972022&ei=In7yXoX0Hba...</td><td>r1---sn-o097znlk.googlevideo...</td><td>video/mp4</td><td>200</td><td>0 B</td><td>1.62 MB</td><td>9.6s</td></tr><tr><td>14</td><td>videoplayback?expire=1592972022&ei=In7yXoX0Hba...</td><td>r1---sn-o097znlk.googlevideo...</td><td>audio/web...</td><td>200</td><td>0 B</td><td>497.58 KB</td><td>2.9s</td></tr><tr><td>15</td><td>videoplayback?expire=1592972022&ei=In7yXoX0Hba...</td><td>r1---sn-o097znlk.googlevideo...</td><td>video/mp4</td><td>200</td><td>0 B</td><td>1.64 MB</td><td>9.8s</td></tr><tr><td>16</td><td>videoplayback?expire=1592972022&ei=In7yXoX0Hba...</td><td>r1---sn-o097znlk.googlevideo...</td><td>video/mp4</td><td>200</td><td>0 B</td><td>1.83 MB</td><td>10.9s</td></tr><tr><td>17</td><td>videoplayback?expire=1592972022&ei=In7yXoX0Hba...</td><td>r1---sn-o097znlk.googlevideo...</td><td>audio/web...</td><td>200</td><td>0 B</td><td>507.89 KB</td><td>3.1s</td></tr><tr><td>18</td><td>videoplayback?expire=1592972022&ei=In7yXoX0Hba...</td><td>r1---sn-o097znlk.googlevideo...</td><td>video/mp4</td><td>200</td><td>0 B</td><td>1.62 MB</td><td>9.6s</td></tr><tr><td>19</td><td>videoplayback?expire=1592972022&ei=In7yXoX0Hba...</td><td>r1---sn-o097znlk.googlevideo...</td><td>audio/web...</td><td>200</td><td>0 B</td><td>458.49 KB</td><td>2.8s</td></tr><tr><td>20</td><td>videoplayback?expire=1592972022&ei=In7yXoX0Hba...</td><td>r1---sn-o097znlk.googlevideo...</td><td>video/mp4</td><td>200</td><td>0 B</td><td>1.74 MB</td><td>10.4s</td></tr><tr><td>21</td><td>videoplayback?expire=1592972022&ei=In7yXoX0Hba...</td><td>r1---sn-o097znlk.googlevideo...</td><td>audio/web...</td><td>200</td><td>0 B</td><td>460.32 KB</td><td>2.7s</td></tr><tr><td colspan="6">Total</td><td>23.84 MB</td><td>149.4s</td></tr><tr><td colspan="6">Grand Total</td><td>23.84 MB</td><td></td></tr><tr><td colspan="6">Duration</td><td></td><td>231.3s</td></tr></table> This sequence reflects that the media data element corresponding to rn 102 (and corresponding range) comprises a digitally encoded portion of the requested program, consistent with the fact that each of the media data elements requested by the client comprise a digitally encoded portion of the program being played.	Overview	Summary	Chart	#	Resource	Host	Mime Type	C...	Header	Body	Time	1	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	video/mp4	200	0 B	1.94 MB	14.6s	2	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	audio/web...	200	0 B	154.89 KB	1.0s	3	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	audio/web...	200	0 B	469.25 KB	3.1s	4	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	video/mp4	200	0 B	1.61 MB	10.3s	5	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	audio/web...	200	0 B	466.26 KB	3.1s	6	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	video/mp4	200	0 B	1.95 MB	12.2s	7	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	audio/web...	200	0 B	482.05 KB	3.1s	8	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	video/mp4	200	0 B	1.96 MB	11.9s	9	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	video/mp4	200	0 B	1.95 MB	12.4s	10	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	audio/web...	200	0 B	475.85 KB	3.2s	11	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	video/mp4	200	0 B	1.64 MB	9.8s	12	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	audio/web...	200	0 B	480.70 KB	2.9s	13	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	video/mp4	200	0 B	1.62 MB	9.6s	14	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	audio/web...	200	0 B	497.58 KB	2.9s	15	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	video/mp4	200	0 B	1.64 MB	9.8s	16	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	video/mp4	200	0 B	1.83 MB	10.9s	17	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	audio/web...	200	0 B	507.89 KB	3.1s	18	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	video/mp4	200	0 B	1.62 MB	9.6s	19	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	audio/web...	200	0 B	458.49 KB	2.8s	20	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	video/mp4	200	0 B	1.74 MB	10.4s	21	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	audio/web...	200	0 B	460.32 KB	2.7s	Total						23.84 MB	149.4s	Grand Total						23.84 MB		Duration							231.3s
Overview	Summary	Chart																																																																																																																																																																																																										
#	Resource	Host	Mime Type	C...	Header	Body	Time																																																																																																																																																																																																					
1	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	video/mp4	200	0 B	1.94 MB	14.6s																																																																																																																																																																																																					
2	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	audio/web...	200	0 B	154.89 KB	1.0s																																																																																																																																																																																																					
3	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	audio/web...	200	0 B	469.25 KB	3.1s																																																																																																																																																																																																					
4	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	video/mp4	200	0 B	1.61 MB	10.3s																																																																																																																																																																																																					
5	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	audio/web...	200	0 B	466.26 KB	3.1s																																																																																																																																																																																																					
6	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	video/mp4	200	0 B	1.95 MB	12.2s																																																																																																																																																																																																					
7	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	audio/web...	200	0 B	482.05 KB	3.1s																																																																																																																																																																																																					
8	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	video/mp4	200	0 B	1.96 MB	11.9s																																																																																																																																																																																																					
9	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	video/mp4	200	0 B	1.95 MB	12.4s																																																																																																																																																																																																					
10	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	audio/web...	200	0 B	475.85 KB	3.2s																																																																																																																																																																																																					
11	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	video/mp4	200	0 B	1.64 MB	9.8s																																																																																																																																																																																																					
12	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	audio/web...	200	0 B	480.70 KB	2.9s																																																																																																																																																																																																					
13	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	video/mp4	200	0 B	1.62 MB	9.6s																																																																																																																																																																																																					
14	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	audio/web...	200	0 B	497.58 KB	2.9s																																																																																																																																																																																																					
15	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	video/mp4	200	0 B	1.64 MB	9.8s																																																																																																																																																																																																					
16	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	video/mp4	200	0 B	1.83 MB	10.9s																																																																																																																																																																																																					
17	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	audio/web...	200	0 B	507.89 KB	3.1s																																																																																																																																																																																																					
18	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	video/mp4	200	0 B	1.62 MB	9.6s																																																																																																																																																																																																					
19	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	audio/web...	200	0 B	458.49 KB	2.8s																																																																																																																																																																																																					
20	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	video/mp4	200	0 B	1.74 MB	10.4s																																																																																																																																																																																																					
21	videoplayback?expire=1592972022&ei=In7yXoX0Hba...	r1---sn-o097znlk.googlevideo...	audio/web...	200	0 B	460.32 KB	2.7s																																																																																																																																																																																																					
Total						23.84 MB	149.4s																																																																																																																																																																																																					
Grand Total						23.84 MB																																																																																																																																																																																																						
Duration							231.3s																																																																																																																																																																																																					
[1.1.4] and having a playback rate,	The requested media data element is digitally encoded at a bitrate range and framerate such that a normal rendition occurs when played back at that framerate, as also made evident by the client player that plays back the media data elements at the corresponding playback rate.																																																																																																																																																																																																											
[1.2.1] receiving each of the requested media data elements via the data connection,	The rows on the left side of the above capture (and in the summary in 1.1.3) reflects 21 successive requests and responses via the data connection (for audio and video segments in the program, which are interleaved), each, like the request/response on the right (for the highlighted row), showing an HTTP “200 OK” response code (“success”). The success code reflects receipt of the requested element by the client.																																																																																																																																																																																																											
[1.2.2] wherein the data connection has a data rate more rapid than the playback rate of the media data elements,	As seen in the overview above, the data for playback is transferred in a fraction of the time required to play it back at normal rendition. It is reasonable to infer, therefore, that the data connection had a data rate in excess of the playback rate.																																																																																																																																																																																																											
[1.2.3] and each received media data element is received at a rate as fast as the data connection between the media	WAG is not aware of any routine in the client-side code that YouTube supplies for the user devices, that in any way throttles the data connection. Furthermore, WAG’s observation is that the timings reflected in these captures for the reception of the																																																																																																																																																																																																											

Claim Language	Contention
source and the media player allows;	individual elements reflect that the data for each element is being received at the full speed of the data connection. Although this limitation is literally met, it can also be found under the doctrine of equivalents. The function of this receiving step is to ensure that the media data elements are received by the user system fast enough to prevent interruptions to the playback on the user device; client devices receive media data elements from YouTube's servers, or the CDN servers under YouTube's control, fast enough to prevent interruptions to playback, as evidenced by the uninterrupted playback on the client end. The way in which this receiving is performed is by receiving the media data elements from the data connection that has a data rate that is more rapid than the playback rate of the media data elements sent via that connection, which, as indicated above in 1.2.2, is what the client does. The result is uninterrupted playing of the time sequenced media data elements on the client device, which, as indicated, is observed on the client device.
[1.3] storing the received media data elements in a memory of the media player;	The client-side code provided by YouTube performs the function of storing each received element in a memory of the media player. This is also confirmed by observing that if the data connection is disrupted (for example, the network cable unplugged on a cabled connection), playback continues for a period of time (but generally not through to the end of the program), reflecting the media data elements were stored in a memory of the media player.
[1.4] playing the received media data elements in series from the memory of the media player; and	The client-side code provided by YouTube performs the function of passing the received elements in series from the player's memory to its playback facility, resulting in the observed continuous and properly time-sequenced playback.
[1.5.1] as the received media data elements are played, the media player automatically sending additional requests for subsequent media data elements for storage in the memory of the media player	See 1.2.1, showing the additional requests for subsequent media elements, which are successively stored as received (see 1.3). The succession of requests is observed to occur automatically (no user intervention after the initial user instruction to Play is required). This step results from the YouTube-provided application (JavaScript or app) making successive requests from a segment listing in the manifest.
[1.5.2] as required to maintain about a predetermined number of media data elements in the memory of the media player during playing.	The client-side code provided by YouTube performs the function of maintaining about a predetermined number of media data elements in the memory of the media player during playing. This may also be seen in the following trace showing the volume and timing of request activity, in the example desktop platform captures from June 2020:

Claim Language	Contention
	A network traffic analysis chart titled 'Overview Summary Chart'. The y-axis represents data rate in MB/s, ranging from 0 to 16,000 in increments of 2,000. The x-axis represents time in seconds, ranging from 0 to 280 in increments of 20. The chart displays two data series: 'Request' (blue bars) and 'Response' (red bars). The 'Request' series shows periodic, low-amplitude spikes, while the 'Response' series shows much higher, periodic spikes that reach approximately 15,000 MB/s. The chart is set to 'Focused' mode, and a 'Settings' button is visible in the top right corner. As can be seen, the player initially makes requests to establish its element store to a desired level, and then periodically makes requests at a rate approximately corresponding to the rate of playback, to maintain the store at the desired level as the media is played out of the store during normal playback. YouTube live captures also reflect a parameter called “rbuf,” which WAG believes is related to the receive buffer of the client. The following trace, taken well into reception of a live stream, reflects the client maintaining an rbuf value in the range of 20,000:

Claim Language	Contention																																																
	<pre> %3D&alr=yes&cpn=YPO-3LLMyPC-Omgj&cver=2.20210202.08.03&headm=3&rm=17&rbuf=0 cver=2.20210202.08.03&headm=3&rm=18&rbuf=0 cver=2.20210202.08.03&sq=5165042&rm=19&rbuf=25008 %3D&alr=yes&cpn=YPO-3LLMyPC-Omgj&cver=2.20210202.08.03&sq=5165042&rm=20&rbuf=25020 cver=2.20210202.08.03&sq=5165043&rm=21&rbuf=8946 %3D&alr=yes&cpn=YPO-3LLMyPC-Omgj&cver=2.20210202.08.03&sq=5165043&rm=22&rbuf=8688 %3D&alr=yes&cpn=YPO-3LLMyPC-Omgj&cver=2.20210202.08.03&sq=5165044&rm=23&rbuf=12695 cver=2.20210202.08.03&sq=5165044&rm=24&rbuf=11438 cver=2.20210202.08.03&sq=5165045&rm=25&rbuf=15954 %3D&alr=yes&cpn=YPO-3LLMyPC-Omgj&cver=2.20210202.08.03&sq=5165045&rm=26&rbuf=15949 %3D&alr=yes&cpn=YPO-3LLMyPC-Omgj&cver=2.20210202.08.03&sq=5165046&rm=27&rbuf=19953 cver=2.20210202.08.03&sq=5165046&rm=28&rbuf=18196 cver=2.20210202.08.03&sq=5165047&rm=29&rbuf=22211 %3D&alr=yes&cpn=YPO-3LLMyPC-Omgj&cver=2.20210202.08.03&sq=5165047&rm=30&rbuf=20958 %3D&alr=yes&cpn=YPO-3LLMyPC-Omgj&cver=2.20210202.08.03&sq=5165048&rm=31&rbuf=24964 cver=2.20210202.08.03&sq=5165048&rm=32&rbuf=24955 cver=2.20210202.08.03&sq=5165049&rm=33&rbuf=24721 %3D&alr=yes&cpn=YPO-3LLMyPC-Omgj&cver=2.20210202.08.03&sq=5165049&rm=34&rbuf=24720 %3D&alr=yes&cpn=YPO-3LLMyPC-Omgj&cver=2.20210202.08.03&sq=5165050&rm=35&rbuf=25191 cver=2.20210202.08.03&sq=5165050&rm=36&rbuf=19189 cver=2.20210202.08.03&sq=5165051&rm=37&rbuf=23932 %3D&alr=yes&cpn=YPO-3LLMyPC-Omgj&cver=2.20210202.08.03&sq=5165051&rm=38&rbuf=19195 cver=2.20210202.08.03&sq=5165052&rm=39&rbuf=23447 %3D&alr=yes&cpn=YPO-3LLMyPC-Omgj&cver=2.20210202.08.03&sq=5165052&rm=40&rbuf=23450 cver=2.20210202.08.03&sq=5165053&rm=41&rbuf=24462 %3D&alr=yes&cpn=YPO-3LLMyPC-Omgj&cver=2.20210202.08.03&sq=5165053&rm=42&rbuf=24455 %3D&alr=yes&cpn=YPO-3LLMyPC-Omgj&cver=2.20210202.08.03&sq=5165054&rm=43&rbuf=24211 cver=2.20210202.08.03&sq=5165054&rm=44&rbuf=20455 cver=2.20210202.08.03&sq=5165055&rm=45&rbuf=24970 %3D&alr=yes&cpn=YPO-3LLMyPC-Omgj&cver=2.20210202.08.03&sq=5165055&rm=47&rbuf=20216 cver=2.20210202.08.03&sq=5165055&rm=48&rbuf=18970 %3D&alr=yes&cpn=YPO-3LLMyPC-Omgj&cver=2.20210202.08.03&sq=5165056&rm=49&rbuf=23471 cver=2.20210202.08.03&sq=5165056&rm=50&rbuf=19212 cver=2.20210202.08.03&sq=5165057&rm=51&rbuf=23682 %3D&alr=yes&cpn=YPO-3LLMyPC-Omgj&cver=2.20210202.08.03&sq=5165057&rm=52&rbuf=19192 cver=2.20210202.08.03&sq=5165058&rm=53&rbuf=23197 %3D&alr=yes&cpn=YPO-3LLMyPC-Omgj&cver=2.20210202.08.03&sq=5165058&rm=54&rbuf=23197 %3D&alr=yes&cpn=YPO-3LLMyPC-Omgj&cver=2.20210202.08.03&sq=5165059&rm=55&rbuf=24452 cver=2.20210202.08.03&sq=5165059&rm=56&rbuf=24440 cver=2.20210202.08.03&sq=5165060&rm=57&rbuf=24454 %3D&alr=yes&cpn=YPO-3LLMyPC-Omgj&cver=2.20210202.08.03&sq=5165060&rm=58&rbuf=24456 cver=2.20210202.08.03&sq=5165061&rm=59&rbuf=24969 %3D&alr=yes&cpn=YPO-3LLMyPC-Omgj&cver=2.20210202.08.03&sq=5165061&rm=60&rbuf=24961 %3D&alr=yes&cpn=YPO-3LLMyPC-Omgj&cver=2.20210202.08.03&sq=5165062&rm=61&rbuf=24967 cver=2.20210202.08.03&sq=5165062&rm=62&rbuf=19963 cver=2.20210202.08.03&sq=5165063&rm=63&rbuf=24432 %3D&alr=yes&cpn=YPO-3LLMyPC-Omgj&cver=2.20210202.08.03&sq=5165063&rm=64&rbuf=20189 cver=2.20210202.08.03&sq=5165064&rm=65&rbuf=24696 </pre> <table> <tr> <th>Name</th><th>Value</th></tr> <tr> <td>expire</td><td>1612703510</td></tr> <tr> <td>ei</td><td>tpIfYM6LHcK6kAP1sYD4Dg</td></tr> <tr> <td>ip</td><td>107.77.205.129</td></tr> <tr> <td>id</td><td>w_Ma8oQLmSM.5</td></tr> <tr> <td>itag</td><td>135</td></tr> <tr> <td>aitags</td><td>133,134,135,136,160</td></tr> <tr> <td>source</td><td>yT_live_broadcast</td></tr> <tr> <td>requiessl</td><td>yes</td></tr> <tr> <td>mh</td><td>wG</td></tr> <tr> <td>mm</td><td>44,26</td></tr> <tr> <td>mn</td><td>sn-n4v7sn76,sn-a5meknzs</td></tr> <tr> <td>ms</td><td>lva,onr</td></tr> <tr> <td>mv</td><td>m</td></tr> <tr> <td>mvi</td><td>3</td></tr> <tr> <td>pl</td><td>24</td></tr> <tr> <td>initswdbps</td><td>273750</td></tr> <tr> <td>vpv</td><td>1</td></tr> <tr> <td>live</td><td>1</td></tr> <tr> <td>hang</td><td>1</td></tr> <tr> <td>noclen</td><td>1</td></tr> <tr> <td>mime</td><td>video/mp4</td></tr> <tr> <td>ns</td><td>pelCcuN6dX8jn76Cq4Wx91YI</td></tr> <tr> <td>gir</td><td>yes</td></tr> </table> HTTP/ Last-Modified: Sun, 07 Content-Type: video/ Date: Sun, 07 Expires: Sun, 07 Cache-Control: private Transfer-Encoding: chunk Alt-Svc: h3=29= X-Walltime: Ms: 161268 X-Bandwidth-Est: 271406 X-Bandwidth-Est-Comp: 967573 X-Bandwidth-Est2: 967573 X-Bandwidth-App-Limited: false X-Bandwidth-Est-App-Limited: false X-Bandwidth-Est3: 756014 X-Sequence-Num: 516506 X-Segment-Lmt: 161268 X-Head-Time-Sec: 257631 X-Head-Time-Millis: 257631 X-Head-Sequnum: 516506 Access-Control-Allow-Origin: https:// Access-Control-Allow-Credentials: true Timing-Allow-Origin: https:// Access-Control-Expose-Headers: Client- Vary: Origin Group: Origin, Response, Referer, ... Headers Text Hex Raw	Name	Value	expire	1612703510	ei	tpIfYM6LHcK6kAP1sYD4Dg	ip	107.77.205.129	id	w_Ma8oQLmSM.5	itag	135	aitags	133,134,135,136,160	source	yT_live_broadcast	requiessl	yes	mh	wG	mm	44,26	mn	sn-n4v7sn76,sn-a5meknzs	ms	lva,onr	mv	m	mvi	3	pl	24	initswdbps	273750	vpv	1	live	1	hang	1	noclen	1	mime	video/mp4	ns	pelCcuN6dX8jn76Cq4Wx91YI	gir	yes
Name	Value																																																
expire	1612703510																																																
ei	tpIfYM6LHcK6kAP1sYD4Dg																																																
ip	107.77.205.129																																																
id	w_Ma8oQLmSM.5																																																
itag	135																																																
aitags	133,134,135,136,160																																																
source	yT_live_broadcast																																																
requiessl	yes																																																
mh	wG																																																
mm	44,26																																																
mn	sn-n4v7sn76,sn-a5meknzs																																																
ms	lva,onr																																																
mv	m																																																
mvi	3																																																
pl	24																																																
initswdbps	273750																																																
vpv	1																																																
live	1																																																
hang	1																																																
noclen	1																																																
mime	video/mp4																																																
ns	pelCcuN6dX8jn76Cq4Wx91YI																																																
gir	yes																																																

Claim Language	Contention
	 This limitation is also met under the doctrine of equivalents. The function of sending additional requests as required to maintain about a predetermined number of media data elements in the memory of the media player during playing is to ensure that the media player does not run out of data, or overflow data, so as to ensure uninterrupted playback of the program, which is evident from the uninterrupted playing of the program on the client device and by the requests discussed above. The way of ensuring that the player memory is not underflowed or overflowed with media data is to monitor the amount of data remaining in the player memory, and/or to monitor available bandwidth between the client and the server, and then balancing anticipated consumption by the client and supply from the server to maintain a level of media data in the player memory that does not underflow or overflow the memory in the media player; this is what happens with the client device, as indicated above by way of both the periodic requests for media data elements and the “rbuf” parameter, both of which show that the client is at least monitoring data remaining in the player memory to ensure there is no underflow or overflow of the memory. The result is that the client obtains a steady stream of

Claim Language	Contention
	media data elements that is sufficient to ensure playback without interruptions, which is evident from the uninterrupted playback on the client device.
2. The method of claim 1, further comprising maintaining in the memory of the media player a record identifying the last media data element received and stored by the media player.	This is reflected, for example, in successive requests, wherein the client is able to recall and increment the identifier of the last media data element received. For example, in June 2020 captured sequence referenced above, the following reflects the requests for rn=104 (and the corresponding range) and the following request for rn=105 (and the corresponding range) (which happen to be for the audio portion of the program):  Similarly, for the video (rn 106 and 108 (and corresponding ranges requested) (the audio and video are being interleaved and rn 107 was audio)):

There are separate sequences of audio and video, and here the client software is keeping track of the last element received for each, in order to be able to request the following segment.

Claim Language	Contention
5. The method of claim 4, wherein the reliable transmission protocol is TCP.	Some of the observed sessions used TCP as the reliable transmission protocol (others used Google QUIC, an alternate reliable transmission protocol).
[6.P] A device comprising a media player operable to receive and play an audio or video program, from a remote media source via a data connection over the Internet,	Google sells devices with built-in applications, which together are operable to receive and play an audio or video program from a remote media source via a data connection over the Internet. Such devices include Google smartphones and tablets, and Google Chromebooks, all of which come with streaming video player apps for streaming from YouTube, preinstalled. Because of encryption of the YouTube streams, and the closed source character of the apps, WAG will seek discovery to verify that, as indicated by external observations, the apps perform in the same infringing manner as the downloaded JavaScript on desktop and iOS devices.
[6.T] the media player comprising	
[6.1] a processor, memory accessible to the processor, and a data connection;	<i>E.g.</i> , Qualcomm Snapdragon processor (phones, tablets), Nvidia Tegra K1 (tablets), Intel Core processors (Chromebooks), 2-16GB RAM, WiFi, LTE connectivity.
[6.2.1] a machine-readable, executable routine containing instructions to cause the processor to send requests from the media player to the media source via the data connection for one or more serially identified media data elements representing an audio or video program, each media data element comprising a digitally encoded portion of the audio or video program and having a playback rate,	<i>E.g.</i> , YouTube app pre-installed on the device, incorporating the same functionality when executed as reported in 1.1.1-1.1.4 and reflected in the evidence recited therein.
[6.2.2] each request specifying one or more serial identifiers of the media data elements requested;	See 1.1.2.

Claim Language	Contention
[6.3] a machine-readable, executable routine containing instructions to cause the media player to receive each of the requested media data elements via the data connection, wherein the data connection has a data rate more rapid than the playback rate of the media data elements, and each received media data element is received at a rate as fast as the data connection between the media source and the media player allows;	See 1.2.2-1.2.3 and 6.2.1.
[6.4] a machine-readable, executable routine containing instructions to cause the processor to store the received media data elements in the memory of the media player;	See 1.3 and 6.2.1.
[6.5] a machine-readable, executable routine containing instructions to cause the received media data elements to be played in series from the memory of the media player; and	See 1.4 and 6.2.1.
[6.6] a machine-readable, executable routine containing instructions to cause the processor to, as the received media data elements are played, automatically send additional requests for subsequent media data elements for storage in the memory of the media player as required to maintain about a predetermined number of media data element in the	See 1.5.1-1.5.2 and 6.2.1.

Claim Language	Contention
memory of the media player during playing.	
7. The device of claim 6, further comprising a machine-readable, executable routine containing instructions to cause the processor to maintain in the memory of the media player a record identifying the last media data element received and stored by the media player.	See 6 and 2
8. The device of claim 6, wherein the serial identifiers are sequential.	See 6 and 3
9. The device of claim 6, wherein the receiving is via a reliable transmission protocol.	See 6 and 4
10. The device of claim 9, wherein the reliable transmission protocol is TCP.	See 9 and 5
[11.P] A computer program product for receiving and playing an audio or video program by a media player having a processor and a memory accessible to the processor, from a remote media source via a data connection over the Internet, the computer program product comprising a non-transitory computer readable storage medium having program instructions embodied therewith,	YouTube makes and uses computer program products as recited in 6.2.1 and Overview above, in the form of JavaScript files stored on its servers for distribution to Internet users of the YouTube Web Sites, and in the form of application software files as pre-loaded in the devices referenced in 6.P, and YouTube has created and stored such computer program products and then provided them to its own as well as third-party app stores including without limitation the YouTube Web Site, as well as Google Play, Apple, and Roku app stores.
[11.T] the program instructions comprising:	
[11.1.1] instructions executable to cause the processor to send requests from the media player to the media source via the	<i>E.g.</i> , the machine instructions in the JavaScript stored on machine-readable media on YouTube's servers for the YouTube Web Sites, and in YouTube apps pre-installed on machine-readable media the devices identified in 6.P, and that are stored and provided to third-

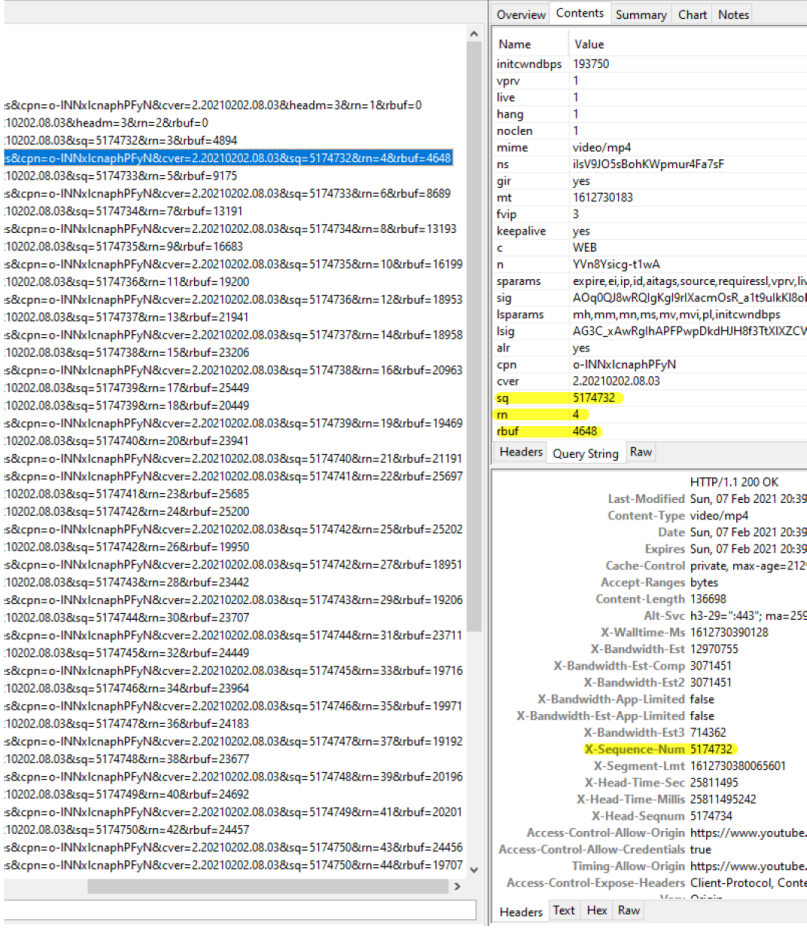
Claim Language	Contention
data connection for one or more serially identified media data elements representing an audio or video program, each media data element comprising a digitally encoded portion of the audio or video program and having a playback rate,	party app stores, incorporating the same functionality when executed as reported in 1.1.1-1.1.4 and reflected in the evidence recited therein.
[11.1.2] each request specifying one or more serial identifiers of the media data elements requested;	See 1.1.2 and 6.2.2.
[11.2] instructions executable to cause the media player to receive each of the requested media data elements via the data connection, wherein the data connection has a data rate more rapid than the playback rate of the media data elements, and each received media data element is received at a rate as fast as the data connection between the media source and the media player allows;	See 1.2.2-1.2.3, 6.2.2, and 6.2.3.
[11.3] instructions executable to cause the processor to store the received media data elements in the memory of the media player;	See 1.3 and 6.2.2.
[11.4] instructions executable to cause the received media data elements to be played in series from the memory of the media player; and	See 1.4.
[11.5] instructions executable to cause the processor to, as the received media data elements are played, automatically send additional	See 1.5.1-1.5.2 and 6.6.

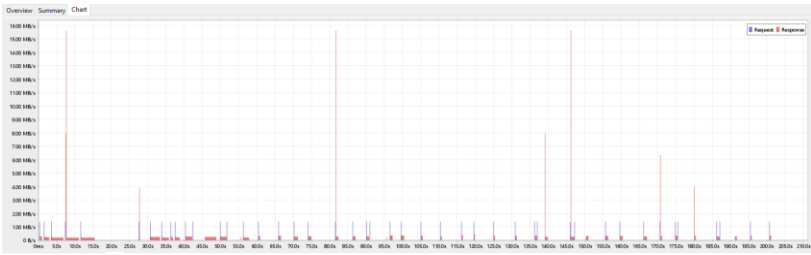
Claim Language	Contention
requests for subsequent media data elements for storage in the memory of the media player as required to maintain about a predetermined number of media data element in the memory of the media player during playing.	
12. The computer program product of claim 11, further comprising instructions executable to cause the processor to maintain in the memory of the media player a record identifying the last media data element received and stored by the media player.	See 11 and 2
13. The computer program product of claim 11, wherein the serial identifiers are sequential.	See 11 and 3
14. The computer program product of claim 11, wherein the computer program product is provided as a software application for the media player.	<i>E.g.</i> , YouTube apps; see 11.P. Includes pre-installed apps as well as apps provided through a Google Play or third-party app store. Also includes YouTube JavaScript (<i>e.g.</i> , base.js) on recorded media made and used by YouTube (<i>e.g.</i> , server-side media, from which the YouTube server reads the JavaScript file for online distribution), which also constitutes a software application for a media player.
15. The computer program product of claim 11, wherein the computer program product is incorporated in the media player.	Applies to the software/firmware in devices supplied with a YouTube app installed. See 6.P and 11.P.
16. The computer program product of claim 11, wherein the receiving is via a reliable transmission protocol.	See 11 and 4
17. The computer program product of claim 16, wherein	See 16 and 5

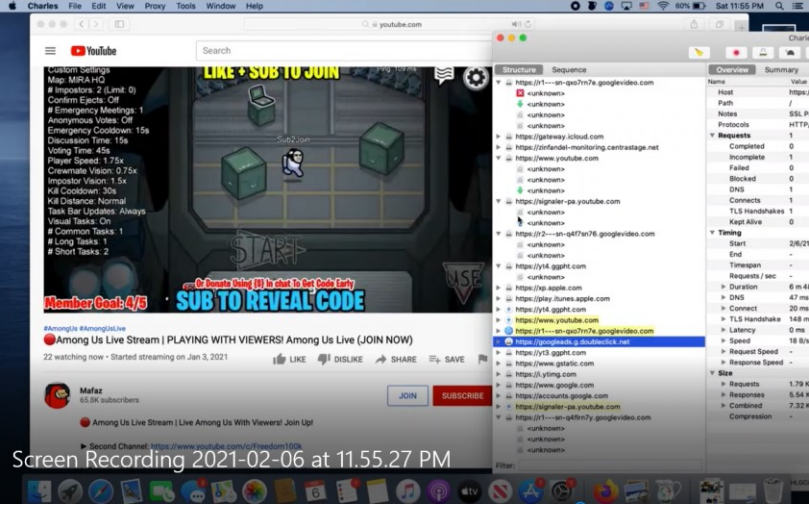
Claim Language	Contention
the reliable transmission protocol is TCP.	

Claim Language	Contention
Overview	WAG has asserted three patents against YouTube, which concern streaming media over the Internet, conducted between a user (client) system, which requests and plays the media, and a server to which the client system connects over the Internet, which provides the requested media. WAG's '824 patent concerns methods, apparatus, and computer program products on the server side of this interaction, for prerecorded media. WAG's '594 patent concerns methods, apparatus, and computer program products on the client side of the above-described interaction. WAG's '636 patent, addressed herein, corresponds to the '824 patent in certain respects, but concerns live media. Because the subject matter of these patents overlap, WAG's infringement contentions with regard to the '824 and '594 patents are each incorporated herein by reference. YouTube LLC and/or affiliated entities such as Google LLC ("YouTube") directs and controls the activity of streaming digitally encoded media it provides over the Internet through the site youtube.com (also accessible as www.youtube.com) (the "YouTube Web Site") from servers owned and/or the operations of which are controlled by YouTube, including servers in the U.S. and in the Western District of Texas ("YouTube's Servers"). These contentions relate to every live streaming video program accessible from or through the YouTube Site (including without limitation the "Live" section of the YouTube Web Site as well as the YouTube TV subscription service), for the period from the respective 2017 issuance dates of the patents addressed in these contentions, through the common date of expiration of the patents (March 28, 2021). They further relate to live videos streamed from YouTube's Servers (e.g., googlevideo.com, tv.youtube.com) to embedded "YouTube" devices and apps. The foregoing services are addressed individually, but are also referred to, for purposes of collective reference, as the "YouTube Streaming Services." YouTube distributes live video programs from YouTube's Servers, over the Internet, to large numbers of users in the U.S. and elsewhere. During the relevant period, this was done via a pull protocol defined at the application layer and generally conforming with the MPEG-DASH specification. (Google is a Charter Member of the DASH Industry Forum.) YouTube's web pages also contain reference to a manifest for HTTP Live Streaming ("HLS"), in addition to providing a link to an MPEG-DASH manifest. HLS is a streaming media delivery specification, compliance with which infringes WAG's patents asserted herein on substantially the same facts shown herein with respect to YouTube's use of MPEG-DASH. The client-side JavaScript files that YouTube distributes in support of its

Claim Language	Contention
	streaming media (<i>e.g.</i>, base.js) also provide functions for HLS. WAG contends that YouTube has supported streaming media delivery during the relevant period via HLS as at least a fallback where the user device identifies itself as a device that normally will not support MPEG-DASH or prefers HLS. YouTube has used its own Google Cloud CDN (which includes edge servers known as Google Global Cache (“GGC”) servers at locations including Midland, El Paso, Austin, and San Antonio, Texas), and also partnered with third-party providers, including without limitation Fastly, Cloudflare, Highwinds, Level 3, and Akamai, for CDN distribution. The pull protocol employed by YouTube is based on HTTP (or HTTPS) requests, which can be responded to through CDN servers at the network Edge that support HTTP/HTTPS retrieval (as most ordinary web servers do), provided the edge server possesses the relevant media data elements. The same streaming mechanisms appear to be consistently supported through each CDN, independent of the type of user-agent on the client side, and in a similar manner over the respective CDNs. The first example shown below, from October 2020, is for a client-side device comprising a Chrome browser on a desktop computer. Insofar as relevant to meeting the limitations below, the evidence is the same in all material respects for all widely used browsers or players (Chrome, Firefox, Internet Explorer, Edge, Safari, etc.) iOS, iPadOS, Android, FireOS, FireTV, Roku, TVOS and other “Smart TV” user systems, regardless of the CDN. The MPEG-DASH form of distribution is reflected in data such as the following, observed through a web debugging proxy inserted in the communication path between a user system accessing a YouTube live video, and the YouTube Server (at a server in the domain googlevideo.com) (The proxy provided is solely for data acquisition and testing purposes and is not in any way necessary to use YouTube’s Streaming Services, nor does it change any of the requests or responses while viewing video). The same request-response patterns may also be readily observed, on desktop and other non-embedded platforms supporting full-featured browsers, through the “developer tools” and like utility functions built into the browsers, for example, by hitting Ctrl-Shift-I and looking at the Network tab during playback. The following is a representative example of the requests and responses observed (in this sample, taken in February 2021 using a proxy) in streaming a live video program from YouTube:

Claim Language	Contention
	 The above trace from a live YouTube transmission shows a request for a media data element identified by an “sq” identifier, 5174732 (as well as an “rn” identifier of 4). The entirety of the portion watched of each requested live video program (from beginning to end of watching) is delivered as a series of time-sequenced media data elements, where each such element has been given one or more serial identifiers (e.g., sq=516042 etc.), and requested by serial identifier, as reflected by the sequence which was recorded and shown here (on the left side of the figure) for a number of successive requests of this type (interleaved for the audio and video segments comprising the stream), as shown above. The client-side software is responsible for requesting each such element, by its serial identifier, when and as needed by the client in order to sustain continuous and uninterrupted playback. Because the streaming media flow as to each element in the stream is thus dependent on specific client-side requests for the respective elements comprising the stream, the overall process is sometimes referred to as a “pull.”

Claim Language	Contention
	Furthermore, in order for the client to successfully keep up with the stream while making such pull requests, it should not take longer to request and receive the individual segments than the time it takes to play the segments back at a normal rendition. Repeated observations confirm that transmissions from YouTube Servers consistently meet these requirements. For example, the following is a graph of data flow for a YouTube live transmission in February 2021, which shows bursts of rapid data transfer in response to requests, followed by quiet periods between the requests, reflecting that the data elements are each being sent in less time than it takes to play them back:  The MPEG-DASH standard, ISO/IEC 23009, spells out media distribution steps that align identically with each of the claim limitations herein. The version of the MPEG-DASH specification referenced herein is the Fourth edition, Dec. 2019. However, WAG believes that YouTube also complied with earlier and later versions of this standard, with no differences between such versions being material to the claims herein. The foregoing figures are representative of every platform streaming live programming from YouTube Servers for which WAG has been able to observe the contents of HTTP(S) traffic, including Chrome, Firefox, Safari, and other desktop browsers, iOS on iPhone and iPad, and Android, as well as TVOS and FireOS. On some embedded platforms (such as Roku and TVOS), due to encryption, WAG has been unable to inspect the contents of HTTPS (encrypted) requests and responses. However, the timing of the encrypted requests and responses follow the same pattern as observable on other platforms (<i>e.g.</i>, desktop Chrome, as well as the FireOS embedded platform), and WAG believes the MPEG-DASH streaming implementation is the same in all material respects to the other observed implementations. WAG will include in its discovery requests inquiries directed to the additional platforms whose communications have been obscured by encryption.

Claim Language	Contention
[1.P.1] A method	This method is performed when live streaming video programming is distributed by YouTube's Servers to users in the U.S.
[1.P.2] for distributing a live audio or video program over the Internet,	The YouTube Streaming Services for live performances are accessible over the Internet through the youtube.com/www.youtube.com website, from YouTube TV, as well as directly from YouTube Servers including googlevideo.com and tv.youtube.com. For example, the following figure shows a game being played, on a live stream (in January 2021):
	 Activating a link on the “Live” section of youtube.com/www.youtube.com, or YouTube TV (or similarly requesting a video through a Roku app or the like) results in a transmission of the streaming media over the Internet for the live performance. YouTube live programming can run at scheduled times, or continuously, <i>e.g.</i>, for a News channel.
[1.P.3] from a server system	YouTube's videos are distributed from servers accessible through youtube.com/www.youtube.com, or directly from servers such as googlevideo.com and tv.youtube.com. See 1.P.2.
[1.P.4] to a plurality of user systems,	The above screen capture reflects a transmission to one user system. Google aims to have a large audience for its live transmissions, and for any given video there are many users similarly situated such that the server system supports a plurality of user systems.

Claim Language	Contention
[1.T] the method comprising:	
[1.1] receiving at the server system a continuous digitally encoded stream for the audio or video program, via a data connection from a live source, in real time, the server system comprising at least one computer;	A POSITA understands that in order to serve live programming, YouTube's servers would necessarily have to perform this claim step, <i>i.e.</i> , receive the stream from a live source, <i>e.g.</i> , a feed from a webcam or a digital feed from a media outlet such as a TV station. WAG contends that this inference, which it does not believe YouTube will dispute, is sufficient, whether YouTube expressly concedes the point or not, to establish that the computers comprising YouTube's server system performs this step.
[1.2] upon receipt of the stream by the server system	Since the object is to provide programming contemporaneously with the performance, WAG contends that it is a reasonable inference that YouTube performs the following steps upon its systems' receipt of the live feed.
[1.2.1] supplying, at the server system, media data elements representing the program,	The figure above, again, but now focusing on the right-hand side, showing the request and corresponding response,

Overview	Contents	Summary	Chart	Notes
Name	Value			
initcwndbps	193750			
vprv	1			
live	1			
hang	1			
noclen	1			
mime	video/mp4			
ns	ilsV9JO5sBohKWpmur4Fa7sF			
gir	yes			
mt	1612730183			
fvip	3			
keepalive	yes			
c	WEB			
n	YVn8Ysicg-t1wA			
sparams	expire,ei,ip,id,aitags,source,requiresl,vprv,li			
sig	AOq0QJ8wRQlgKgl9rlXacmOsR_a1t9ulkKI8ol			
lsparams	mh,mm,mn,ms,mv,mvi,pl,initcwndbps			
lsig	AG3C_xAwRglhAPFPwpDkdHJH8f3TtXIXZCV			
alr	yes			
cpn	o-INNxIcnaphPFyN			
cver	2.20210202.08.03			
sq	5174732			
rn	4			
rbuf	4648			
Headers	Query String	Raw		
HTTP/1.1 200 OK Last-Modified Sun, 07 Feb 2021 20:39 Content-Type video/mp4 Date Sun, 07 Feb 2021 20:39 Expires Sun, 07 Feb 2021 20:39 Cache-Control private, max-age=212 Accept-Ranges bytes Content-Length 136698 Alt-Svc h3-29=":443"; ma=256 X-Walltime-Ms 1612730390128 X-Bandwidth-Est 12970755 X-Bandwidth-Est-Comp 3071451 X-Bandwidth-Est2 3071451 X-Bandwidth-App-Limited false X-Bandwidth-Est-App-Limited false X-Bandwidth-Est3 714362 X-Sequence-Num 5174732 X-Segment-Lmt 1612730380065601 X-Head-Time-Sec 25811495 X-Head-Time-Millis 25811495242 X-Head-Seqnum 5174734 Access-Control-Allow-Origin https://www.youtube. Access-Control-Allow-Credentials true Timing-Allow-Origin https://www.youtube. Access-Control-Expose-Headers Client-Protocol, Conte Main-Origin				
Headers	Text	Hex	Raw	

Claim Language	Contention
	shows one user request and corresponding server response, for one media data element in a sequence of many such elements (each similarly requested and provided) comprising the requested program stream (e.g., the entire requested video, which can go on for many minutes or in many cases continuously), from beginning to end of the portion being watched. The media data element requested in the above figure is one of a plurality of media data elements representing the requested program. WAG does not have access to direct proof that YouTube supplies, at its server system, media data elements corresponding to these requests. However, a POSITA understands that in order to serve the media data elements comprising the requested program, and have them received, as indicated by the “200 OK” “success” HTTP response code, they each necessarily must be supplied at the server system. The manifest noted above also shows that the server supplies the media data elements representing the program. WAG contends that this inference, based on an observed series of user requests for such elements, and corresponding responses, proximate to each other in time, which it does not believe YouTube will dispute, is sufficient, whether YouTube expressly concedes the point or not, to establish that the computers comprising YouTube’s server system perform the step, under YouTube’s direction and control, of supplying each requested element to the server in order for WAG to have been able to observe and record the corresponding responses. The manifest for a live YouTube program (showing the most recent entries) also reflects that the YouTube server informs the client system that streaming media data elements representing the program have been supplied at the server: <pre> </SegmentList> [Many preceding sequential entries like the below ...] <SegmentURL media="sq/2698308/lmt/1636893105779885"/> <SegmentURL media="sq/2698309/lmt/1636893105779900"/> <SegmentURL media="sq/2698310/lmt/1636893105779915"/> <SegmentURL media="sq/2698311/lmt/1636893105779930"/> <SegmentURL media="sq/2698312/lmt/1636893105779945"/> <SegmentURL media="sq/2698313/lmt/1636893105779960"/> <SegmentURL media="sq/2698314/lmt/1636893105779975"/> <SegmentURL media="sq/2698315/lmt/1636893105779990"/> <SegmentURL media="sq/2698316/lmt/1636893105780005"/> </SegmentList> </pre>

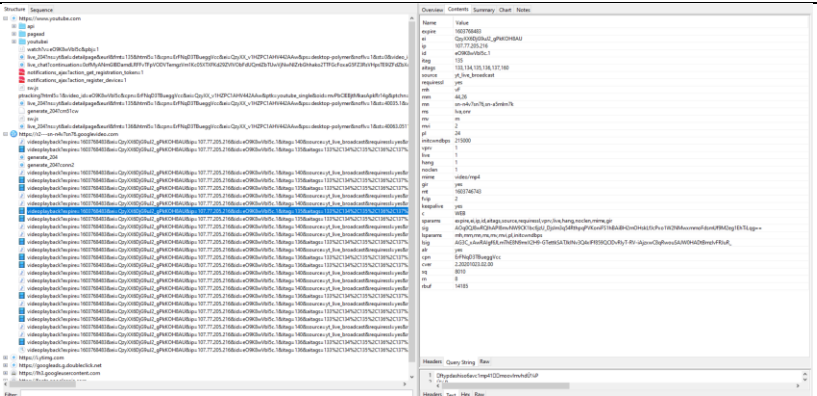
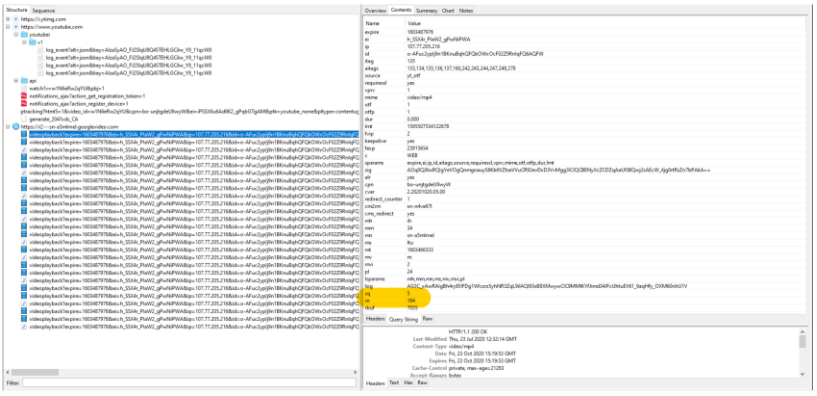
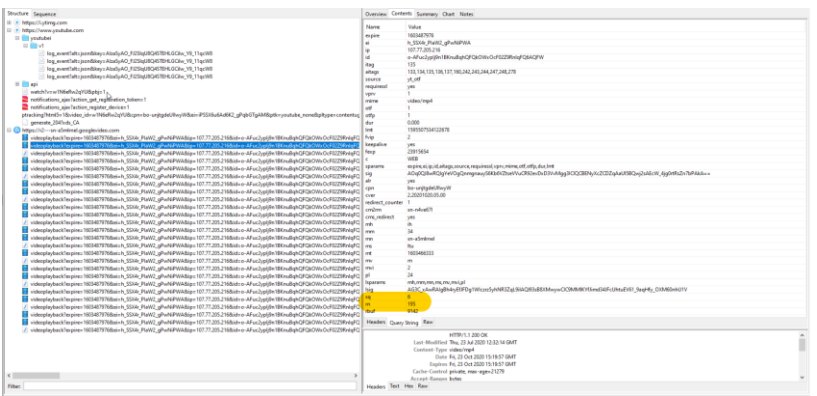
Claim Language	Contention
	where the string “sq/[number]/lmt/[number]” given for each element represents a path to where the element is stored on the server. (WAG believes that the appended “lmt” number is a timestamp, used to avoid playing outdated segments.)
[1.2.2] each media data element comprising a digitally encoded portion of the program	In the above example (sq = 5174732), the requested element comprises 136,698 bytes, representing a portion (one of a large number requested over the course of receiving the program) of the requested program. The metadata shown (200 OK) reflects that the element received is what was requested, corresponding to the request parameters (including sq and rn). This represents one portion of the program, as reflected by the timeline on the left of the below figure for the series of requests that included the above individual request (highlighted).

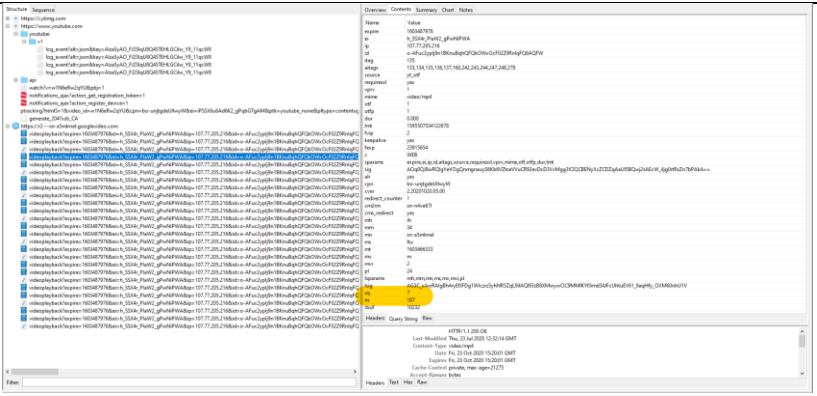
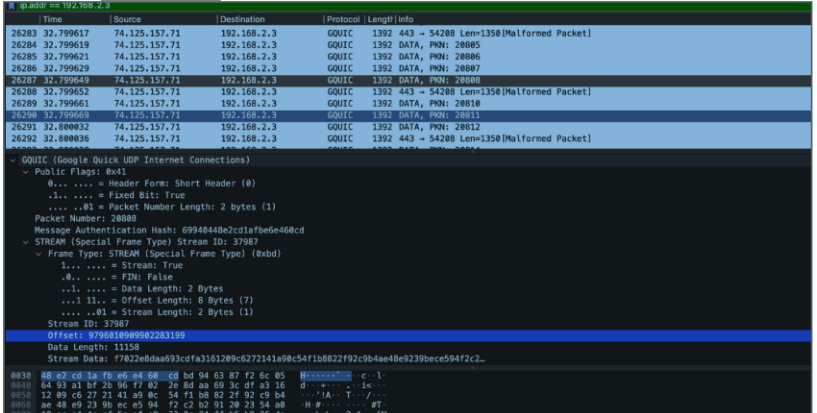
Claim Language	Contention
	:s&cpn=o-INNXIcnaphPFyN&cver=2.20210202.08.03&headm=3&rn=1&rbuf=0 :10202.08.03&headm=3&rn=2&rbuf=0 :10202.08.03&sq=5174732&rn=3&rbuf=4894 :s&cpn=o-INNXIcnaphPFyN&cver=2.20210202.08.03&sq=5174732&rn=4&rbuf=4648 :10202.08.03&sq=5174733&rn=5&rbuf=9175 :s&cpn=o-INNXIcnaphPFyN&cver=2.20210202.08.03&sq=5174733&rn=6&rbuf=8689 :10202.08.03&sq=5174734&rn=7&rbuf=13191 :s&cpn=o-INNXIcnaphPFyN&cver=2.20210202.08.03&sq=5174734&rn=8&rbuf=13193 :10202.08.03&sq=5174735&rn=9&rbuf=16683 :s&cpn=o-INNXIcnaphPFyN&cver=2.20210202.08.03&sq=5174735&rn=10&rbuf=16199 :10202.08.03&sq=5174736&rn=11&rbuf=19200 :s&cpn=o-INNXIcnaphPFyN&cver=2.20210202.08.03&sq=5174736&rn=12&rbuf=18953 :10202.08.03&sq=5174737&rn=13&rbuf=21941 :s&cpn=o-INNXIcnaphPFyN&cver=2.20210202.08.03&sq=5174737&rn=14&rbuf=18958 :10202.08.03&sq=5174738&rn=15&rbuf=23206 :s&cpn=o-INNXIcnaphPFyN&cver=2.20210202.08.03&sq=5174738&rn=16&rbuf=20963 :10202.08.03&sq=5174739&rn=17&rbuf=25449 :10202.08.03&sq=5174739&rn=18&rbuf=20449 :s&cpn=o-INNXIcnaphPFyN&cver=2.20210202.08.03&sq=5174739&rn=19&rbuf=19469 :10202.08.03&sq=5174740&rn=20&rbuf=23941 :s&cpn=o-INNXIcnaphPFyN&cver=2.20210202.08.03&sq=5174740&rn=21&rbuf=21191 :s&cpn=o-INNXIcnaphPFyN&cver=2.20210202.08.03&sq=5174741&rn=22&rbuf=25697 :10202.08.03&sq=5174741&rn=23&rbuf=25685 :10202.08.03&sq=5174742&rn=24&rbuf=25200 :s&cpn=o-INNXIcnaphPFyN&cver=2.20210202.08.03&sq=5174742&rn=25&rbuf=25202 :10202.08.03&sq=5174742&rn=26&rbuf=19950 :s&cpn=o-INNXIcnaphPFyN&cver=2.20210202.08.03&sq=5174742&rn=27&rbuf=18951 :10202.08.03&sq=5174743&rn=28&rbuf=23442 :s&cpn=o-INNXIcnaphPFyN&cver=2.20210202.08.03&sq=5174743&rn=29&rbuf=19206 :10202.08.03&sq=5174744&rn=30&rbuf=23707 :s&cpn=o-INNXIcnaphPFyN&cver=2.20210202.08.03&sq=5174744&rn=31&rbuf=23711 :10202.08.03&sq=5174745&rn=32&rbuf=24449 :s&cpn=o-INNXIcnaphPFyN&cver=2.20210202.08.03&sq=5174745&rn=33&rbuf=19716 :10202.08.03&sq=5174746&rn=34&rbuf=23964 :s&cpn=o-INNXIcnaphPFyN&cver=2.20210202.08.03&sq=5174746&rn=35&rbuf=19971 :10202.08.03&sq=5174747&rn=36&rbuf=24183 :s&cpn=o-INNXIcnaphPFyN&cver=2.20210202.08.03&sq=5174747&rn=37&rbuf=19192 :10202.08.03&sq=5174748&rn=38&rbuf=23677 :s&cpn=o-INNXIcnaphPFyN&cver=2.20210202.08.03&sq=5174748&rn=39&rbuf=20196 :10202.08.03&sq=5174749&rn=40&rbuf=24692 :s&cpn=o-INNXIcnaphPFyN&cver=2.20210202.08.03&sq=5174749&rn=41&rbuf=20201 :10202.08.03&sq=5174750&rn=42&rbuf=24457 :s&cpn=o-INNXIcnaphPFyN&cver=2.20210202.08.03&sq=5174750&rn=43&rbuf=24456 :s&cpn=o-INNXIcnaphPFyN&cver=2.20210202.08.03&sq=5174750&rn=44&rbuf=19707 Further in the same stream, the following was recorded:

Claim Language	Contention
	 Hence, each of these media data elements comprises a digitally encoded portion of the program.
[1.2.3] and having a playback rate;	The requested media data element is digitally encoded at a bitrate range and framerate such that a normal rendition occurs when played back at that framerate, <i>i.e.</i> , when each is played back in the corresponding time period set out in the above listing.
[1.3.1] serially identifying the media data elements,	The elements in this example are serially identified by parameters including “sq” and “rn” as specified in the request headers and query string (and as reflected in the manifest, for sq). Such identification takes place at least when the data element is stored on a storage device accessible to the server, when a manifest is created, and when the server responds to a request. The “sq” and “rn” values, both individually and as an ordered pair, literally provide serial identification of the media data elements. Nonetheless, this limitation may also be found under the doctrine of equivalents. In particular, the function of serially identifying the media data elements is to uniquely identify the time-sequenced

Claim Language	Contention
	media data elements from each other, which is exactly what the “sq” and “rn” values do, both individually and as an ordered pair. The way in which the media data elements are uniquely identified is to provide each with a unique identifier that can be used in requests from the client to the server; the unique “sq” and “rn” values associated with each media data element perform this function, both individually and as an ordered pair. The result of the serial identification is that the client and server agree upon identifications that allow the client to uniquely request a media data element in the proper time sequence, which is what occurs when the client uses the “sq” and/or “rn” values to specify a desired media data element.
[1.3.2] said serial identification indicating a time sequence of the media data elements, and	The time-sequenced character of the successive media data elements in a transmission is reflected in the timeline of requests, see 1.2.2. above. The sq and rn numbers generally increment by 1 for each request. The same can also be directly observed (without the intermediate proxy) on a full-featured browser used to access the YouTube Streaming Services, by invoking its Developer Tools (typically, Ctrl-Shift-I). The above pattern, showing a continuous time sequence of elements comprising the requested stream, corresponding to the serial identification of the elements, is representative of all request-response sequences observed with the YouTube Streaming Services, regardless of the time of observation during the period from patent issuance through March 2021, regardless of the CDN being utilized, and regardless of the user media playing platform. In many cases (as shown here) the audio is part of the numbering. In all cases, the range numbers also progress serially. The following are corresponding request-response timelines for other performances and platforms streaming from YouTube Streaming Services: NBC News (Live) October 2020:

Page 13 of 24
Petitioners' Exhibit 1047
Page 0060

Claim Language	Contention
	
	Desktop captures from July 2000: 
	

Claim Language	Contention
	 Note that in both of these examples, sq and rn are both serial, sq is sequential for video and audio (in separate sequences) and rn is sequential for audio + video. WAG has also been able to intercept successive requests and responses of characteristic length for streaming YouTube media to tvOS devices, but encryption has prevented WAG from seeing the identifiers exchanged. For example, a packet capture for a YouTube streaming session from tvOS showed the following:  WAG believes these embedded platforms (e.g., tvOS, FireOS, etc., as well as Roku during the relevant period) operated with YouTube in substantially the same way as observed for desktop and iOS devices. WAG will include in its discovery requests inquiries directed to the additional platforms whose communications have been obscured by encryption. Additionally, the list of media data elements in the manifest noted above, each uniquely identified by its respective “sq” number, also provides serial identification indicating the time sequences of the respective media data elements. Hence, not only is time sequencing of the media data elements indicated by progressive

Claim Language	Contention
	increasing of the “sq” and/or “rn” values, but also by the relative positioning of the identified media data element within the manifest, with media data elements appearing higher (or first) in the manifest being earlier in time.
[1.4] storing the media data elements in a data structure under the control of the server system;	As noted above, the elements in streaming media requests to YouTube’s Streaming Services are serially identified by serial identifiers including “sq” and “rn” as specified in the request and response. WAG cannot directly see how YouTube stores the media data elements comprising its live programs in data structures on (or under the control of) its servers. However, WAG believes that, since the requests are successful, that corresponding elements (which conform to the requested sq and rn parameters) are stored on the server. Moreover, as discussed above, the captured manifest data also reflects that the media data elements are stored on the server in a folder/file data structure (or in an equivalent layout in a database or other data structure), wherein the folder containing the corresponding segment is a subfolder named with the sq number, situated under a parent folder named “sq”. (There is also with an appended “lmt” subfolder, which WAG believes represents a timestamp.)
[1.5.1] receiving requests at the server system via one or more data connections over the Internet,	The diagrams above (both the requests and responses, and the associated timeline) show that the intermediate proxy intercepted requests being made from a user system over Internet data connections, to YouTube servers, and time-proximate response thereto, reflecting the same serial identification as the corresponding request, with the return code 200 OK. WAG contends it is therefore reasonable to infer that the YouTube server received the request.
[1.5.2] for one or more of the media data elements stored in the data structure,	See examples in 1.3.2. The shown GET requests for the element identified by serial identifiers, including “sq” and “rn”, from a specified resource (in the URL) reflects receiving a request for one or more media data elements stored in the data structure. The MPEG-DASH specification also provides for the client to request one or more media data elements stored in a data structure on the server, and the observed YouTube manifests (see example above) confirm compliance with that aspect of the specification.
[1.5.3] each received request specifying one or more serial identifiers of the requested one or more media data elements,	See 1.5.2

Claim Language	Contention
[1.5.4] each received request originating from a requesting user system of a plurality of user systems; and	The intercepting proxy (or alternately the browser's Developer Tools console) verifies that the request came a requesting user system.
[1.6.1] responsive to the requests, sending, by the server system, the one or more media data elements having the one or more specified serial identifiers,	Reflected in all of the responses intercepted by the intermediate proxy, as shown in the various diagrams herein, and directly reflected in Developer Tools windows on the browsers that support such tools. The result code 200 OK shown in the captures confirm success in this regard.
[1.6.2] to the requesting user systems corresponding to the requests;	The responses go the requesting user systems corresponding to the requests; see 1.6.1. The response can be viewed (as data) in the Developer Console, as well as rendered on the user device display and audio output.
[1.W] wherein	
[1.W.1] the data connection between the server system and each requesting user system has a data rate more rapid than the playback rate of the one or more media data elements sent via that connection;	As seen with respect to 1.3.2, requests are made every few seconds (reflecting that the elements corresponding to the element serial ID take the corresponding number of seconds to play back), whereas each element is generally transmitted in well under a second. It is reasonable to infer, therefore, that the data connection had a data rate in excess of the playback rate.
[1.W.2] each sending is at a transmission rate as fast as the data connection between the server system and each requesting user system allow;	WAG does not have access to YouTube's servers, but states that is not aware of, and sees no evidence of, any process on YouTube's servers, or any routine in the client-side code that YouTube supplies for the user devices, that in any way throttles the data connection. WAG contends that it is therefore reasonable to infer that as a result of the observed time sequence of requests and responses, and the inferred playback duration of each of the elements, that each sending is at a transmission rate as fast as the data connection between the server system and each requesting user system allow. Although this limitation is literally met, it can also be found under the doctrine of equivalents. The function of this sending step is to ensure that the media data elements are received by the user system fast enough to prevent interruptions to the playback on the user device; YouTube's servers, or the CDN servers under YouTube's control, send the media data elements to the client devices fast enough to prevent interruptions to playback, as evidenced by the uninterrupted playback on the client end. The way in which this sending is performed is by providing the media

Claim Language	Contention
	data elements to the data connection between the server system and each requesting user system that has a data rate that is more rapid than the playback rate of the media data elements sent via that connection, which, as indicated above in 1.W.1, YouTube's servers, or the CDN's under YouTube's control, also do. The result is uninterrupted playing of the time sequenced media data elements on the client device, which, as indicated, is observed on the client device.
[1.W.3] the one or more media data element sent are selected without depending on the server system maintaining a record of the last media data element sent to the requesting user systems;	As reflected in the data shown herein, the observed requests for media data elements comprising a program all originate on the client side and the responses thereto are proximate in time to the corresponding request. Further, there is no observed communication from the server to the requesting user system to indicate the last media data element that the server sent to the requesting user system. Each client request specifies a fully-specified ID that is not relative to an earlier requested ID. Furthermore, the MPEG-DASH standard states "Clients may request data using the HTTP protocol from standard web servers that have no DASH-specific capabilities." ISO/IEC 23009 at ix, 11. WAG contends that is therefore reasonable to infer that the only data transmissions from the server of the program being streamed are those responsive to specific requests (by serial identifier), and that, for lack of any observed communication from the server as to the last media data element that it sent, the client's requests are made as determined on the client side, with no dependency on any record maintained by the server system of the last media data element sent, and the server's responses, each keyed to the corresponding request, likewise have no dependency on any record of the server system as to the last media data element that it sent. Although met literally, this limitation may also be found under the doctrine of equivalents. The function of the server selecting the media data elements without dependence on the server maintaining a record of the last media data element being sent is to allow the client to control the pacing and delivery of the media data elements rather than the server. As set forth above, it is the client that controls pacing and delivery of the media data elements from the YouTube servers (or the CDNs under YouTube's control). The way in which this occurs is to allow the client to issue requests for media data elements uniquely identified by serial identifiers and for the server to respond to these requests with the corresponding media data elements. As set forth above, this is what YouTube's servers (or the CDN's under YouTube's control) do. The result is that the client primarily controls what media data elements are sent and received, and when. As indicated above, this is what happens with YouTube's streaming service.

Claim Language	Contention
[1.W.4] all of the media data elements that are sent by the server system to the one or more user systems are sent in response to the requests; and	WAG has observed no transmission of streaming media data from any of the YouTube Streaming Services other than those in direct response to user-side requests by serial ID, as discussed throughout the foregoing. The streaming media received, from beginning to end of the program, is 100% via pull, with each media data element being sent in response to a request therefor.
[1.W.5] all of the media data elements that are sent by the server system to the requesting user systems are sent from the data structure under the control of the server system as the media data elements were first stored therein.	See 1.4. WAG contends that the media data elements are stored on YouTube's servers as reflected in the client requests, as indicated in a manifest, and that the media data elements so structured are served to the requesting user systems as first stored.
2. The method of claim 1 wherein the serial identifiers are sequential.	As reflected in the data reproduced herein, YouTube's serial identifiers are generally also sequential. There are numerous examples of sequential runs of identifiers.
3. The method of claim 1, wherein the sending is via a reliable transmission protocol.	The combination of HTTP and TLS as reflected in the data herein indicates the use of a reliable transmission protocol.
4. The method of claim 3, wherein the reliable transmission protocol is TCP.	Some of the observed sessions used TCP as the reliable transmission protocol (others used Google QUIC, an alternate reliable transmission protocol).
[5.P] A server system for distributing a live audio or video program over the Internet to a plurality of user systems, the server system comprising:	See 1.P.1; the claim here is for the server that performs the method of claim 1.
[5.1] at least one computer having a connection to the Internet;	YouTube's server is reachable over the Internet and it is therefore reasonable to infer that it comprises at least one computer having a connection to the Internet.

Claim Language	Contention
[5.2] a machine-readable, executable routine containing instructions to cause one of the at least one computers to receive a continuous digitally encoded stream for the live audio or video program, via a data connection from a live source, in real time,	See 1.1.
[5.3] a machine-readable, executable routine containing instructions to cause one of the at least one computers, upon receipt of the stream by the server system,	See 1.2.
[5.3.1] to supply, at the server system, media data elements representing the program, each media data element comprising a digitally encoded portion of the program and having a playback rate,	See 1.2.1-1.2.3.
[5.3.2] to serially identify the media data elements, said serial identification indicating a time sequence of the media data elements, and	See 1.3.1-1.3.2.
[5.3.3] to store the media data elements in a data structure under the control of the server system;	See 1.4.
[5.4] a machine-readable, executable routine containing instructions to cause one of the at least one computers to receive requests at the server system via one or more data connections over the Internet, for one or more of the media data elements stored in the data structure, each received request specifying one or more serial identifiers of the one or more media data elements, each received request originating from a requesting	See 1.5.1-1.5.4.

Claim Language	Contention
user system of a plurality of user systems; and	
[5.5] a machine-readable, executable routine containing instructions to cause one of the at least one computers to send, responsive to the requests, the one or more media data elements having the one or more specified serial identifiers, to the requesting user systems corresponding to the requests; wherein	See 1.6.1-1.6.2.
[5.W.1] the data connection between the server system and each requesting user system has a data rate more rapid than the playback rate of the one or more media data elements sent via that connection;	See 1.W.1.
[5.W.2] each sending is at a transmission rate as fast as the data connection between the server system and each requesting user system allow;	See 1.W.2.
[5.W.3] the one or more media data elements send are selected without depending on the server system maintaining a record of the last media data element sent to the requesting user systems;	See 1.W.3.
[5.W.4] all of the media data elements that are sent by the server to the one or more user systems are sent in response to the requests; and	See 1.W.4.
[5.W.5] all of the media data elements that are sent by the server system to the requesting user systems are sent from the data structure under the control of the server system as the media data elements were first stored therein.	See 1.W.5.

Claim Language	Contention
6. The server system of claim 5 wherein the serial identifiers are sequential.	See 5 and 2.
7. The server system of claim 5, wherein the sending is via a reliable transmission protocol.	See 5 and 3.
8. The server system of claim 7, wherein the reliable transmission protocol is TCP.	See 7 and 4.
[9.P] A computer program product for distributing a live audio or video program over the Internet from a server system comprising at least one computer to a plurality of user systems, the computer program product comprising a non-transitory computer readable storage medium having program instructions embodied therewith, the program instructions comprising:	See 1.P.1; the claim here is for a computer program/non-transitory media embodying program instructions that implement the method of claim 1.
[9.1] instructions executable to cause one of the at least one computers to receive a continuous digitally encoded stream for the audio or video program, via a data connection from a live source, in real time;	See 1.1.
[9.2] instructions executable to cause one of the at least one computers, upon receipt of the stream by the server system,	See 1.2.
[9.2.1] to supply, at the server system, media data elements representing the program, each media data element comprising a digitally encoded portion of the program and having a playback rate,	See 1.2.1-1.2.3.

Claim Language	Contention
[9.2.2] to serially identify the media data elements, said serial identification indicating a time sequence of the media data elements, and	See 1.3.1-1.3.2.
[9.2.3] to store the media data elements in a data structure under the control of the server system;	See 1.4.
[9.2.4] instructions executable to cause one of the at least one computers to receive requests at the server system via one or more data connections over the Internet, for one or more of the media data elements stored in the data structure, each received request specifying one or more serial identifiers of the one or more media data elements, each received request originating from a requesting user system of a plurality of user systems; and	See 1.5.1-1.5.4.
[9.2.5] instructions executable to cause one of the at least one computers to send, responsive to the requests, the one or more media data elements having the one or more specified serial identifiers to the requesting user systems corresponding to the requests; wherein	See 1.6.1-1.6.2.
[9.W.1] the data connection between the server system and each requesting user systems has a data rate more rapid than the playback rate of the one or more media data elements sent via that connection;	See 1.W.1.
[9.W.2] each sending is at a transmission rate as fast as the data connection between the	See 1.W.2.

Claim Language	Contention
server system and each requesting user system allows;	
[9.W.3] the one or more media data elements sent are selected without depending on the server system maintaining a record of the last media data element sent to the requesting user systems;	See 1.W.3.
[9.W.4] all of the media data elements that are sent by the server to the one or more user systems are sent in response to the requests; and	See 1.W.4.
[9.W.5] all of the media data elements that are sent by the server system to the requesting user systems are sent from the data structure under the control of the server system as the media data elements were first stored therein.	See 1.W.5.
10. The computer program product of claim 9 wherein the serial identifiers are sequential.	See 9 and 2.
11. The computer program product of claim 9, wherein the sending is via a reliable transmission protocol.	See 9 and 3.
12. The computer program product of claim 11, wherein the reliable transmission protocol is TCP.	See 11 and 4.