Bruce S. Sostek, admitted *pro hac vice*
bruce.sostek@tklaw.com
Richard L. Wynne, Jr., admitted *pro hac vice*
richard.wynne@tklaw.com
Adrienne E. Dominquez, admitted *pro hac vice*
adrienne.dominguez@tklaw.com
THOMPSON & KNIGHT LLP
One Arts Plaza
1722 Routh Street, Suite 1500
Dallas, Texas 75201
Telephone: (214) 969-1700
Facsimile: (214) 969-1751

John V. Picone III, Bar No. 187226
jpicone@hopkinscarley.com
Christopher A. Hohn, Bar No. 271759
chohn@hopkinscarley.com
HOPKINS & CARLEY
A Law Corporation
The Letitia Building
70 South First Street
San Jose, CA 95113-2406

mailing address:
P.O. Box 1469
San Jose, CA 95109-1469
Telephone: (408) 286-9800
Facsimile: (408) 998-4790

Attorneys for Plaintiffs
BROADCOM CORPORATION and AVAGO TECHNOLOGIES INTERNATIONAL SALES PTE. LIMITED.

UNITED STATES DISTRICT COURT

CENTRAL DISTRICT OF CALIFORNIA

SOUTHERN DIVISION

BROADCOM CORPORATION and AVAGO TECHNOLOGIES INTERNATIONAL SALES PTE. LIMITED, Plaintiffs, v. NETFLIX, INC., Defendant.	Case No. 8:20-cv-00529-JVS-ADS **FIRST AMENDED COMPLAINT FOR PATENT INFRINGEMENT** **DEMAND FOR JURY TRIAL**

Plaintiffs Broadcom Corporation ("Broadcom Corp.") and Avago Technologies International Sales Pte. Limited ("Avago") (collectively, the "Broadcom Entities") file this First Amended Complaint for Patent Infringement against Defendant Netflix, Inc. ("Netflix") and allege as follows:

NATURE OF THIS ACTION

1. This complaint alleges patent infringement. The Broadcom Entities allege that Netflix has infringed and continues to infringe, directly and/or indirectly, twelve patents: U.S. Patent Nos. 7,266,079 (the "'079 Patent"); 8,259,121 (the "'121 Patent"); 8,959,245 (the "'245 Patent"); 8,270,992 (the "'992 Patent"); 6,341,375 (the "'375 Patent"); 8,572,138 (the "'138 Patent"); 6,744,387 (the "'387 Patent"); 6,982,663 (the "'663 Patent"); 9,332,283 (the "'283 Patent"), 8,548,976 (the "'976 Patent"); 7,457,722 (the "'722 Patent"); and 8,365,183 (the "'183 Patent"). Copies of these patents (collectively, the "Patents-in-Suit") are attached hereto as **Exhibits A-L.**

2. The Patents-in-Suit cover foundational technologies that are essential to various aspects of Netflix's video streaming service, and the systems that Netflix uses to support this service.

3. Netflix directly infringes the Patents-in-Suit by making, using, offering to sell, selling, and/or importing into the United States internet video streaming technology, software, and services that practice the inventions claimed in the Patents-in-Suit. Netflix directs and controls each relevant aspect of the accused technology discussed herein, and benefits from the use of each feature that infringes the Patents-in-Suit.

4. Netflix indirectly infringes the Patents-in-Suit by inducing its

consumer end-users to directly infringe these patents. For example, Netflix induces infringement by providing software (e.g., the Netflix application) that, when used by consumers or other content viewers to stream digital content to televisions, personal computers, phones, tablets, video game consoles, and other devices, as directed and intended by Netflix, causes those end-users to use and practice the inventions claimed in the Patents-in-Suit.

5. The Broadcom Entities seek damages and other relief for Netflix's infringement of the Patents-in-Suit.

PARTIES

6. Plaintiff Broadcom Corporation is a California corporation headquartered at 1320 Ridder Park Drive, San Jose, California 95131. Broadcom Corp. maintains offices within the Central District of California at 15101 Alton Parkway, Irvine, California 92618. Broadcom Corp. is an indirect subsidiary of Broadcom, Inc.

7. Plaintiff Avago Technologies International Sales Pte. Ltd. is a corporation formed under the laws of Singapore with places of business at 1320 Ridder Park Dr., San Jose, California 95131 and 1 Yishun Avenue 7, Singapore 768923. Avago is also an indirect subsidiary of Broadcom, Inc.

8. Defendant Netflix, Inc. is a Delaware corporation that maintains its principal place of business and global headquarters at 100 Winchester Circle, Los Gatos, 95032.

9. Netflix maintains regular and established places of business in this District, including an office at 5808 Sunset Blvd., Los Angeles, CA 90028, where Netflix employs hundreds of people. According to Netflix's website, the Los

Angeles office "is the entertainment hub for Netflix with teams such as Content, Legal, Marketing & Publicity and is located on the Sunset Bronson Studio Lot where a variety of Netflix content is created."[1]

10. Netflix may be served through its registered agent for service of process in California: CT Corporation System, 818 W. Seventh St, Suite 930, Los Angeles, CA 90017.

11. Netflix claims to be a global leader in streaming digital video content. Netflix streams videos of various types, such as films and television series, to over 180 million paid members in over 190 countries. Upon information and belief, Netflix designs, operates, tests, manufactures, uses, offers for sale, sells, and/or imports into the United States—including in the Central District of California—internet video streaming software, systems, and services that generate billions of dollars of revenue for Netflix each year.

JURISDICTION AND VENUE

12. The Broadcom Entities bring this civil action for patent infringement under the Patent Laws of the United States, 35 U.S.C. § 1 et. seq., including 35 U.S.C. §§ 271, 281-285. This Court has subject matter jurisdiction over this action pursuant to 28 U.S.C. §§ 1331 and 1338.

13. The Broadcom Entities' claims for relief arise, at least in part, from Netflix's business contacts and other activities in the State of California and in this District. Upon information and belief, Netflix has committed acts of infringement within this District and the State of California by making, using, selling, offering for sale, and/or importing into the United States and this District products, systems,

[1] https://jobs.netflix.com/locations/los-angeles-california.

and services that infringe one or more claims of the Patents-in-Suit as set forth herein. Further, Netflix induces others within this District to infringe one or more claims of the Patents-in-Suit.

14. Venue is proper in this district and division under 28 U.S.C. §§ 1391(b)-(d) and 1400(b) because Netflix has committed acts of infringement in the Central District of California and has a regular and established physical place of business in Los Angeles, part of the Central District. Upon information and belief, Netflix employs engineers and technical professionals of many disciplines at its Los Angeles facility.

FACTUAL BACKGROUND

15. Henry Samueli and Henry Nicholas founded Broadcom in 1991 in Los Angeles, California. Since then, Broadcom has grown to be a global technology company that produces category-leading semiconductor and infrastructure software solutions. Among other things, Broadcom provides one of the industry’s broadest portfolios of highly integrated semiconductor chips that seamlessly deliver voice, video, data, and multimedia connectivity in the home, office, and mobile environments. From its headquarters in San Jose, California, Broadcom has expanded its footprint across the United States and around the world, employing thousands of individuals globally and in the United States. An overview of Broadcom’s history can be found on its website at: https://www.broadcom.com/company/about-us/company-history/.

16. Broadcom’s continued success depends in substantial part upon its constant attention to research and development. Broadcom and its subsidiaries spend billions of dollars on research and development for their products each year.

Because of this focus, Broadcom has produced a wide range of novel technologies and inventions that are directed to advancements in, among other things, semiconductor design and digital communications, digital content distribution, enterprise and data center networking, home connectivity, set top boxes, infrastructure software, and other technologies integral to business and consumer settings across the United States and throughout the world.

17. Broadcom relies on the patent system as an important part of its intellectual property program to protect the valuable technology and inventions resulting from its research and development efforts. The Broadcom Entities and their related entities have tens of thousands of patents in the United States and abroad.

18. In addition to their internally developed inventions and associated intellectual property, the Broadcom group of companies have acquired technology and intellectual property through mergers and acquisition with other major technology companies, such as the Avago family of companies, LSI, Brocade, CA, Inc. (formerly known as Computer Associates International, Inc.), and Symantec's enterprise business.

19. As explained in detail below, Netflix has built its familiar video streaming business, in part, on the Broadcom Entities' patented technology. Netflix relies on this technology for crucial aspects of the Netflix streaming service. This includes, for example, the Netflix systems used to ensure effective and reliable delivery of streaming content with minimal interruptions, to ensure the efficient, effective use of Netflix server resources, and to encode Netflix streaming content in a format compatible with a large percentage of the client devices (e.g., computers,

smart TVs, mobile phones, and videogame consoles) used to access the Netflix service.

20. In doing so, Netflix has caused, and continues to cause, substantial and irreparable harm to the Broadcom Entities. For instance, the Broadcom Entities sell semiconductor chips used in the set top boxes that enable traditional cable television services. Upon information and belief, as a direct result of the on-demand streaming services provided by Netflix, the market for traditional cable services that require set top boxes has declined, and continues to decline, thereby substantially reducing Broadcom's set top box business.

21. For instance, it is widely reported that the rise of on-demand video streaming services such as Netflix has concurrently lead to a decrease in demand for traditional cable services. As an example, *Variety* reported in February 2019 that "[t]he five biggest U.S. pay-television providers saw their traditional subscriber rolls shrink 4.2% in 2018, as they collectively lost around 3.2 million customers for the year. That's an acceleration from estimated sector-wide declines of 3.7% in 2017 and 2% in 2016." The article attributes the loss in part to a migration of customers to Netflix.[2]

22. Upon information and belief, Netflix could not displace traditional cable television services, or could not do so as effectively, without the use of the Broadcom Entities' patented technology, which—as explained above—enable critical aspects of Netflix's systems.

[2] https://variety.com/2019/biz/news/cord-cutting-2018-accelerate-us-pay-tv-subscribers-1203138404/.

23. Netflix is aware of the Broadcom Entities' patent portfolio, including specifically most of the patents asserted in this Complaint, based on communications between Netflix and the Broadcom Entities. Representatives of the Broadcom Entities have repeatedly attempted to engage Netflix in licensing discussions. As part of these attempts, the Broadcom Entities informed Netflix of its infringement of most of the patents asserted in this Complaint[3] on or about September 26, 2019, and the parties engaged in in-person discussions on October 24, 2019. Netflix did not dispute the infringement presentations the Broadcom Entities provided to Netflix, or otherwise assert that it did and does not infringe the patents identified to Netflix. Unfortunately, Netflix declined to agree to terms for a license for its use of the Broadcom Entities' patents and technology, and declined to present a counteroffer to license terms offered by the Broadcom Entities.

24. Left with no other choice, the Broadcom Entities bring this action to protect their rights and their investment in the research and development of novel technologies.

FIRST CLAIM FOR RELIEF

(Infringement of U.S. Patent No. 7,266,079)

25. The Broadcom Entities reallege and incorporate by reference the allegations of paragraphs 1-24 set forth above.

26. The '079 Patent, entitled "Dynamic Network Load Balancing Over Heterogeneous Link Speed," was duly and legally issued on September 4, 2007 from a patent application filed on July 2, 2001, with Kan Frankie Fan as the named inventor. A copy of the '079 Patent is attached hereto as **Exhibit A**.

[3] These discussions did not address the '283, '976, '722, and '183 Patents.

27. The ’079 Patent claims priority from U.S. Provisional Application No. 60/233,338, filed on September 18, 2000.

28. The ’079 Patent was assigned to Avago, which currently holds all substantial rights, title, and interest in and to the ’079 Patent.

29. Pursuant to 35 U.S.C. § 282, the ’079 Patent is presumed valid.

30. The ’079 Patent is directed to an improvement in the functionality of networked computer systems by “balancing data flow there through.”[4] Specifically, the ’079 Patent’s claims describe a new approach for balancing transmission unit traffic over the multiple heterogeneous links that often connect computing platforms in a computer network.

31. The ’079 Patent addresses a specific technical problem that arose in the computer networking environment as the networks grew ever larger and more complex, and as users sought to transmit ever greater volumes of data across these networks. As the ’079 Patent states, “[t]he present invention relates to communications apparatus and methods, particularly to computer networking apparatus and methods, and more particularly to computer networking apparatus and methods for balancing data flow there through.”[5]

32. As the ’079 Patent explains, “[a] common problem in communication networks is maintaining efficient utilization of network resources, particularly with regard to bandwidth, so that data traffic is efficiently distributed over the available links between sources and destinations.”[6] “Prior art solutions include apparatus and methods that balance data traffic over homogeneous (same-speed) links between

[4] ’079 Patent, 1:17-21.

[5] *Id.*

[6] *Id.* at 1:23-27.

heterogeneous or homogeneous computing platforms (servers, clients, etc.)."[7] However, "[i]ncreasingly, high-performance computing platforms communicate with other computers, routers, switches, and the like, using multiple links which, for a variety of reasons, may operate at disparate link speeds."[8] "For example…adverse network conditions may degrade the performance of one or more links, effectively presenting a heterogeneous-link-speed environment to the server and its link partner(s)."[9]

33. Accordingly, a need existed for a means to "dynamically balance transmission unit traffic in a heterogeneous-link-speed environment" in order to improve the functionality of computer networks.[10]

34. The ’079 Patent claims specific, novel ways to solve these technical problems by dynamically balancing data traffic in a computer networking environment with heterogeneous link speeds. The claims of the ’079 Patent are directed to new, improved methods and apparatuses for balancing transmission unit traffic over networks links.

35. The methods and apparatuses described in the ’079 Patent improve the functionality of a networked computer system by balancing the data traffic among network links having different speeds, capabilities, and congestion levels that connect the various networked elements, thereby improving the speed and efficiency of data transmission within the network.

36. Claim 1 of the ’079 Patent reads as follows:

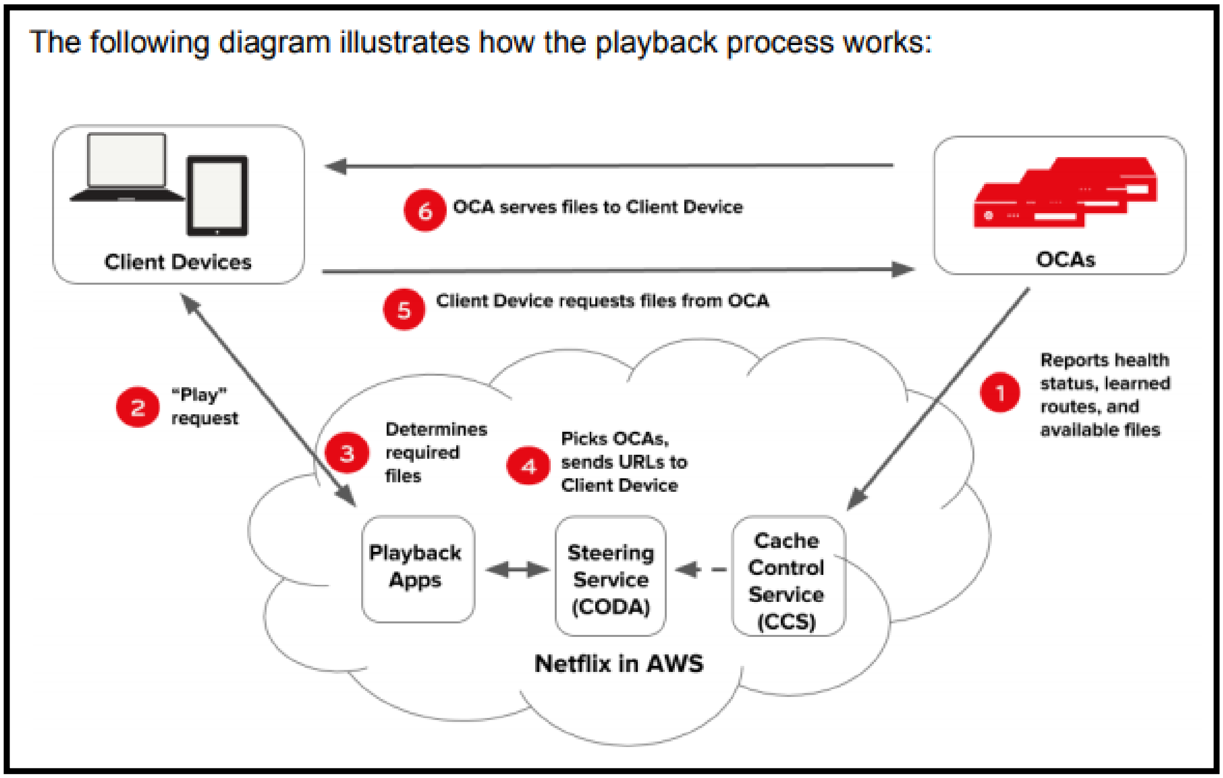
[7] *Id.* at 1:27-30.
[8] *Id.* at 1:30-34.
[9] *Id.* at 1:34-40.
[10] *Id.* at 1:40-43.

> A method for balancing transmission unit traffic over network links, comprising:
>
> a. disposing transmission units into flows;
>
> b. grouping flows into first flow lists, each of the first flow lists corresponding to a selected network link;
>
> c. determining a traffic metric representative of a traffic load on the selected network link;
>
> d. responsive to the traffic metric, regrouping flows into second flow lists corresponding to the selected network link, the regrouping balancing the transmission unit traffic among the network links; and
>
> e. transmitting the respective second flow list over the respective selected network link.

37. Netflix directly infringes the ‘079 Patent by making, using, offering to sell, and/or selling in the United States its Netflix service, which utilizes the inventions claimed in the ‘079 Patent to balance traffic over Netflix’s systems, including its content delivery network (“CDN”).

38. Netflix directly infringes at least independent claim 1 of the ’079 Patent at least in the exemplary manner described below.

39. Netflix utilizes the claimed “method for balancing transmission unit traffic over network links,” including, for instance, in operating its CDN, which Netflix uses to stream video content to its subscribers over the internet. The Netflix CDN is illustrated in the following diagram.



The following diagram illustrates how the playback process works:

Client Devices
6 OCA serves files to Client Device
5 Client Device requests files from OCA
OCAs
1 Reports health status, learned routes, and available files
2 "Play" request
3 Determines required files
4 Picks OCAs, sends URLs to Client Device
Playback Apps
Steering Service (CODA)
Cache Control Service (CCS)
Netflix in AWS

Source: https://openconnect.netflix.com/Open-Connect-Overview.pdf.

40. The Netflix CDN comprises hardware that Netflix builds and operates. Additionally, for some aspects of its CDN, Netflix uses hardware produced and maintained by third parties, including cloud computing services purchased from third parties. However, all the infringing technologies discussed in this Complaint are developed and controlled by Netflix, including though the use of Netflix software that controls the relevant functions performed by the underlying hardware.

41. Notably, in apparent recognition that its cloud computing services could be used by a customer to infringe any number of patents involving a computer as part of a larger system or process, the AWS standard customer agreement makes clear that AWS has no liability for patent infringement claims

arising from infringement by combinations of AWS's services with any other product, service, software, data, content or method.[11]

42. As part of its CDN, Netflix created, operates, uses, and maintains a "global network of thousands of [Open Connect Appliances]," also known as OCAs.[12]

43. As Netflix explains, "[t]he building blocks of Open Connect are our suite of purpose-built server appliances, called Open Connect Appliances (OCAs). These appliances store and serve our video content, with the sole responsibility of delivering playable bits to client devices as fast as possible."[13]

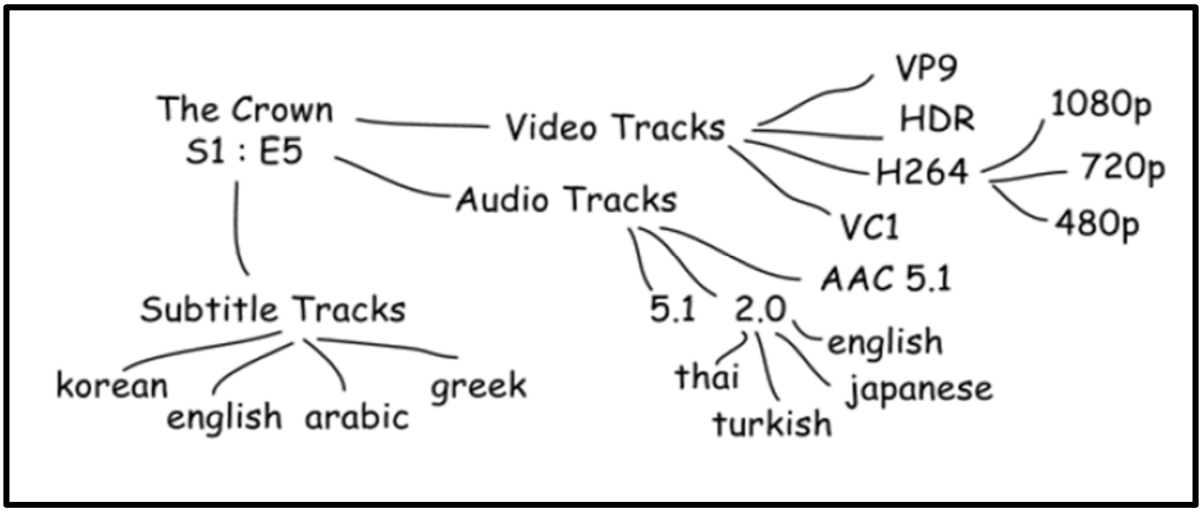
44. Within the CDN, Netflix stores the TV programs and movies that it offers as a series of files. Netflix's customers access Netflix's video content through different types of client devices, including digital televisions, desktop computers, laptop computers, tablet computers, and mobile phones. These client devices are produced by many different manufacturers. Each device has certain capabilities and features that require media to be delivered in a specific form or format. In many cases, the media format used by one device cannot be used by another. Thus, Netflix must make its content available to its users in many different formats.

45. In Netflix's words:

> Every title is encoded in multiple formats, or *encoding profiles*. For example, some profiles may be used by iOS devices and others for a certain class of Smart TVs. There are video profiles, audio profiles, and profiles that contain subtitles. Each audio and video profile is encoded

[11] https://aws.amazon.com/agreement/.
[12] https://openconnect.netflix.com/Open-Connect-Overview.pdf.
[13] *Id.*

> into different levels of quality. For a given title, the higher the number of bits used to encode a second of content (bps), the higher the quality….Finally, we have audio profiles and subtitles available in multiple languages. So for each quadruple of (title, encoding profile, bitrate, language), we need to cache one or more files. As an example, for streaming one episode of The Crown we store around 1,200 files!"[14]



Source: https://medium.com/netflix-techblog/content-popularity-for-open-connect-b86d56f613b.

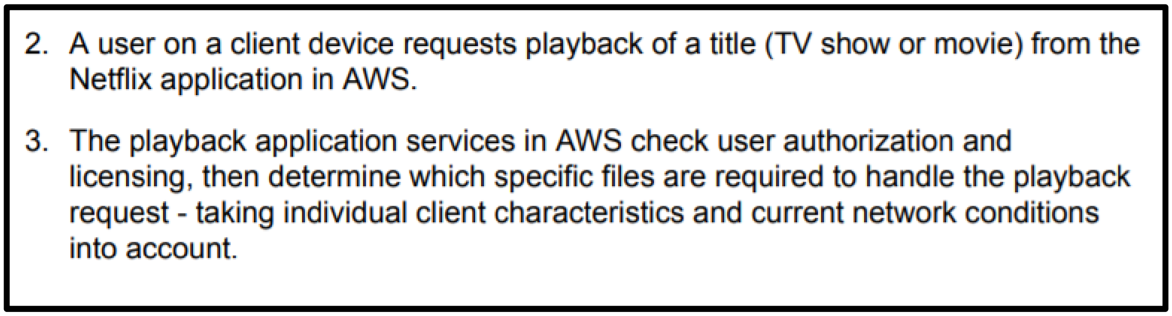
46. Netflix divides these various content files "into 2 second intervals called *segments*."[15]

47. When a Netflix subscriber initiates a playback session by selecting a movie or TV show and pressing "play," the Netflix system "determine[s] which specific files are required to handle the playback request—taking individual client characteristics and current network conditions into account." The Netflix system "pick[s] OCAs that the requested files should be served from [and] generates URLs

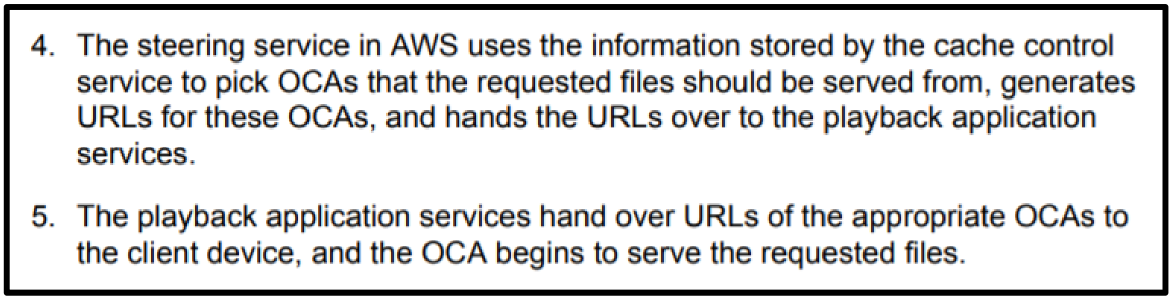
[14] https://medium.com/netflix-techblog/content-popularity-for-open-connect-b86d56f613b.

[15] https://cs.uwaterloo.ca/~brecht/papers/iiswc-netflix-wload-2016.pdf.

for these OCAs…" The Netflix system then "hand[s] over URLs of the appropriate OCAs to the client device, and the OCA begins to serve the requested files."



2. A user on a client device requests playback of a title (TV show or movie) from the Netflix application in AWS.
3. The playback application services in AWS check user authorization and licensing, then determine which specific files are required to handle the playback request - taking individual client characteristics and current network conditions into account.



4. The steering service in AWS uses the information stored by the cache control service to pick OCAs that the requested files should be served from, generates URLs for these OCAs, and hands the URLs over to the playback application services.
5. The playback application services hand over URLs of the appropriate OCAs to the client device, and the OCA begins to serve the requested files.

Source: https://openconnect.netflix.com/Open-Connect-Overview.pdf.

48. Additionally, when a Netflix subscriber initiates a playback session, the Netflix system provides the "offsets of all segments."[16] The client device, under the control of the Netflix application running on it, then "downloads segment offsets and content from multiple content files with different bit rates, then selects a starting bitrate that can be supported by available network bandwidth."[17] The client device "continue[s] to download segments sequentially from the same file unless network or server conditions change (which may result in switching to a different bit rate) or a user event occurs (e.g., stopping or skipping to a new title position)."[18] Thus, and as described further below, Netflix performs a "method for balancing transmission unit traffic over network links."

[16] https://cs.uwaterloo.ca/~brecht/papers/iiswc-netflix-wload-2016.pdf.
[17] *Id.*
[18] *Id.*

49. The Netflix CDN also "dispos[es]" the "transmission units into flows." For example, each of the above-described "segments" of a given content file constitutes a "transmission unit." The Netflix system "disposes" these segments "into flows" by causing the transmission of a series of segments of various content files—each a "flow"—to each of its many subscribers during the playback process.

50. Upon information and belief, the Netflix system also "group[s] flows into first flow lists," each of which "correspond[] to a selected network link." For example, each of the OCAs, and each of the file locations specified by a URL within each OCA, constitutes a "network link." As explained above, Netflix "group[s] flows into first flow lists" by picking the OCAs, and the specific files thereon, that are used to serve a subscriber's playback request for each of the thousands, or even millions, of subscribers streaming video content from Netflix at any given moment.

51. On information and belief, the Netflix system "determin[es] a traffic metric representative of a traffic load on the selected network link."

52. For example, during the playback process, which is controlled by Netflix software, the client device "intelligently selects which OCA to use."[19] "It does this by testing the quality of the network connection to each OCA. It will connect to the fastest, most reliable OCA first."[20] "The client keeps running these tests throughout the video streaming process. The client probes to figure out the best way to receive content from the OCA."[21] Thus, through the playback process,

[19] http://highscalability.com/blog/2017/12/11/netflix-what-happens-when-you-press-play.html.
[20] *Id.*
[21] *Id.*

the Netflix system "determin[es] a traffic metric representative of a traffic load on the selected network link."

53. Upon information and belief, "responsive to the traffic metric," the Netflix system "regroup[s] flows into second flow lists corresponding to the selected network link." For example, over the course of the playback session, the Netflix system redirects flows of content to different OCAs and to different URLs on the same OCA (i.e., different "network links"), thereby regrouping the original set of flows from a given OCA or URL to various Netflix subscribers to a second flow list corresponding to a different OCA or URL.

54. The "regrouping balanc[es] the transmission unit traffic upon the network links." This element is illustrated, for example, by Figure 6B of the '079 Patent.

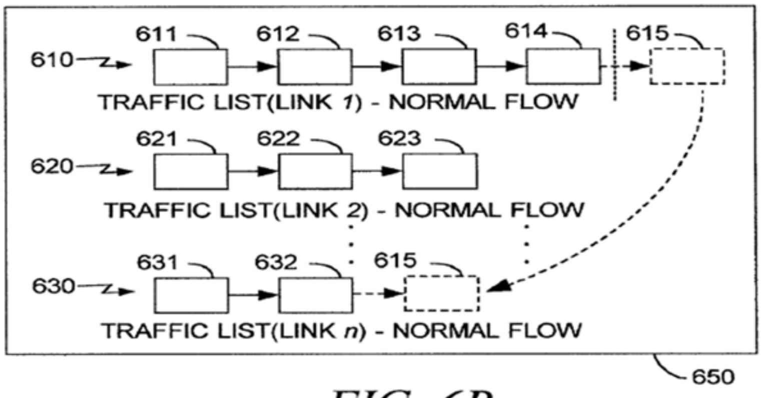

610
611
612
613
614
615
TRAFFIC LIST(LINK 1) - NORMAL FLOW
620
621
622
623
TRAFFIC LIST(LINK 2) - NORMAL FLOW
630
631
632
615
TRAFFIC LIST(LINK n) - NORMAL FLOW
650

FIG. 6B

Source: '079 Patent Specification, Fig. 6B.

55. For instance, the Netflix system moves flows of content files from one OCA or URL to another less congested or otherwise more favorable OCA or URL.

56. Finally, the Netflix system "transmit[s] the respective second flow list over the respective selected network link." For instance, after the Netflix system

transitions a flow of a particular content files to a new OCA or URL, it continues to transmit all the flows then associated with the new OCA or URL (i.e., the “second flow list”) to the many Netflix subscribers who are streaming from that OCA or URL at that moment.

57. At least as of on or around September 26, 2019, when the Broadcom Entities informed Netflix of its infringement of the ’079 Patent, and by no later than the filing and service of the original complaint in this matter, Netflix has had knowledge of the ’079 Patent and Netflix’s infringement thereof.

58. Netflix also has induced, and continues to induce, direct infringement by third-parties of at least claim 1 of the ’079 Patent, at least in the exemplary manner described above, by actively encouraging their use of the Netflix system, in violation 35 U.S.C. § 271(b).

59. For example, Netflix has induced, and continues to induce, direct infringement of the ’079 Patent by customers and/or end users of client devices enabled with the Netflix software application and service. In light of the notice the Broadcom Entities provided to Netflix of its infringement of the ’079 Patent, Netflix knows that it provides and specifically intends to provide an application for use on client devices that, when used as intended with the Netflix streaming service, meets the limitations of claim 1 of the ’079 Patent. Netflix knows and specifically intends that its end users practice the method recited in claim 1 of the ’079 Patent, when using its application and service as intended.

60. Netflix’s knowing and willful infringement of the ’079 Patent has caused and continues to cause damage to Avago, and Avago is entitled to recover

damages sustained as a result of Netflix's wrongful acts in an amount subject to proof at trial.

SECOND CLAIM FOR RELIEF

(Infringement of U.S. Patent No. 8,259,121)

61. The Broadcom Entities reallege and incorporate by reference the allegations of paragraphs 1-60 set forth above.

62. The '121 Patent, entitled "System and Method for Processing Data Using a Network," was duly and legally issued on September 4, 2012 from a patent application filed on December 9, 2002, with Patrick Law, Darren Neuman, and David Baer as the named inventors. A copy of the '121 Patent is attached hereto as **Exhibit B**.

63. The '121 Patent claims priority from U.S. Provisional Application No. 60/420,151, filed on December 9, 2002.

64. The '121 Patent is assigned to Avago, which currently holds all substantial rights, title, and interest in and to the '121 Patent.

65. Pursuant to 35 U.S.C. § 282, the '121 Patent is presumed valid.

66. The claims in the '121 Patent are directed to an improved network for processing audio and visual ("A/V") data. Specifically, the inventions described in the '121 Patent "relate[] to a network environment in an A/V system using 'A/V decoders,' where the A/V decoders are adapted to process, decode or decompress one or more input data streams."[22]

67. The '121 Patent addresses a technical problem in a network processing A/V data. The patent explains that, at the time, there was "no known

[22] '121 Patent, 1:42-45.

methodological way to connect video processing modules in A/V systems" and that "[m]ost video processing modules are connected together in an ad hoc manner."[23] "As a result, such ad-hoc designs may become difficult to verify, maintain and reuse. Furthermore, as more features are added to the A/V systems…it becomes more difficult to design and integrate such features properly."[24] Thus, there was "a need for an architecture or network that provides a general model illustrating how various video processing modules behave in a network environment."[25]

68. The '121 Patent describes and claims solutions to these technical problems, including specific, novel networks with various features for processing A/V data.

69. The inventions described and claimed in the '121 Patent have applications both in the context of the ecosystem within a computer or device, and in more complex computer networking environments. Indeed, as the patent explains, "[m]any modifications and variations of the present invention are possible in light of the above teachings. Thus, it is to be understood that, within the scope of the appended claims, the invention may be practiced otherwise than as described hereinabove."[26]

70. The systems and methods described in the '121 Patent improve the functionality of computer networks used for processing A/V data by providing a new, advantageous approach for those networks.

71. Claim 1 of the '121 Patent is directed to:

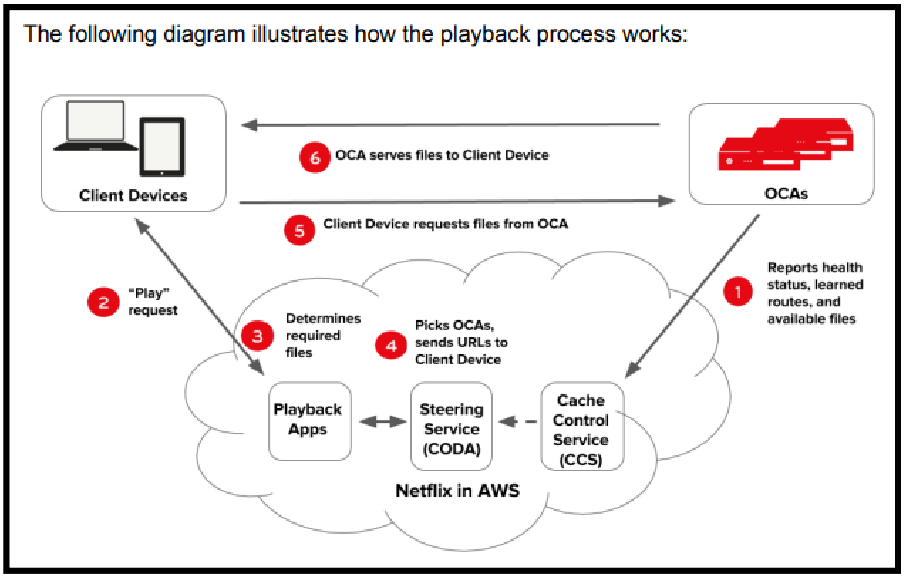
[23] *Id.* at 1:48-51.
[24] *Id.* at 1:51-55.
[25] *Id.* at 1:66-2:1.
[26] *Id.* at 16:26-30.

> A network for processing data configured by a controller to form at least one display pipeline therein by dynamically selecting use of at least two selectable nodes from a plurality of selectable nodes and dynamically concatenating the selected at least two selectable nodes in the network together, wherein said at least one display pipeline has an independent data rate and a flow control module enables said independent data rate.

72. Netflix directly infringes the ’121 Patent by making, using, offering to sell, and/or selling in the United States its Netflix service, which utilizes Netflix’s CDN to process audio and visual data in a manner that uses the inventions claimed in the ’121 Patent.

73. Upon information and belief, Netflix directly infringes at least independent claim 1 of the ’121 Patent at least in the exemplary manner described below.

74. Netflix created, operates, and maintains a “network for processing data,” namely, the CDN that Netflix uses to stream TV shows and movies to its subscribers over the internet, as illustrated in the following diagram from Netflix’s website.



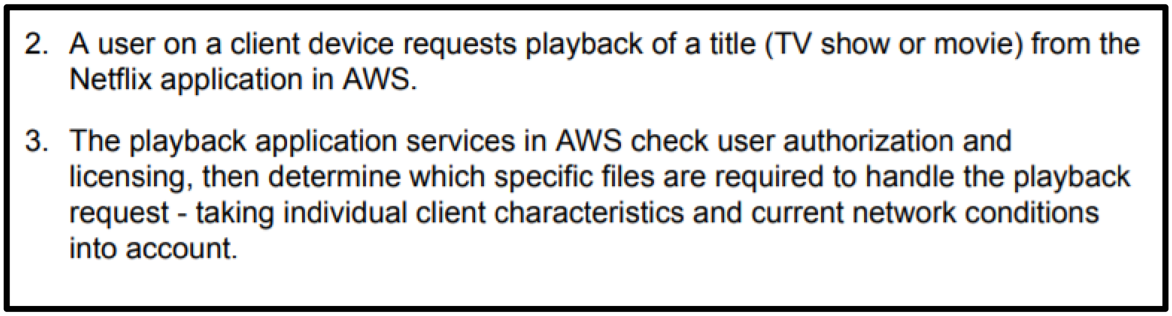
The following diagram illustrates how the playback process works:

Client Devices
6 OCA serves files to Client Device
OCAs
5 Client Device requests files from OCA
2 “Play” request
1 Reports health status, learned routes, and available files
3 Determines required files
4 Picks OCAs, sends URLs to Client Device
Playback Apps
Steering Service (CODA)
Cache Control Service (CCS)
Netflix in AWS

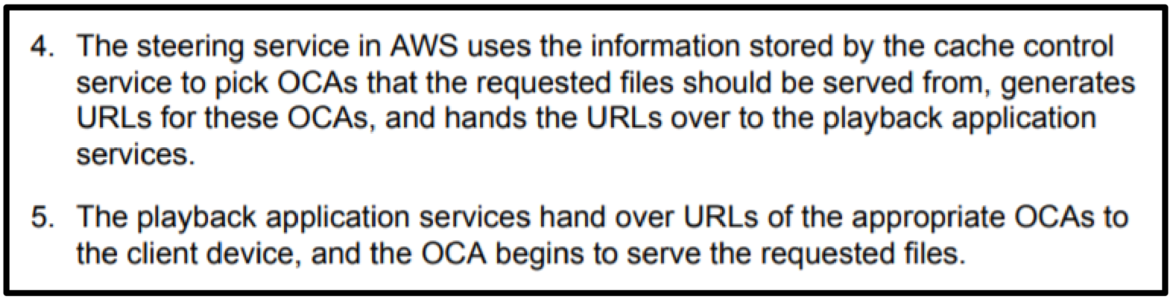
Source: https://openconnect.netflix.com/Open-Connect-Overview.pdf.

75. As shown in this diagram, the Netflix CDN consists of three primary groups of systems: (1) the client devices (e.g., smart TVs, computers, mobile phones, etc.) that subscribers use to access the Netflix service with the help software applications that were developed, or partially developed, by Netflix; (2) the Netflix "backend" (referred to in the diagram as "Netflix in AWS"), which receives and processes requests for video content from subscribers; and (3) a network of OCAs, which deliver the video content to the client devices.

76. The CDN is "configured by a controller." For example, when a Netflix user requests playback of a particular title (e.g., a film or television program) using a client device, various computing resources in the Netflix backend, which are controlled by Netflix and run Netflix applications: (1) receive the request; (2) determine which specific streaming assets are required to handle the request, taking individual client characteristics and current network conditions into account; (3) pick OCAs from which the requested streaming assets should be streamed; and (4) provide a manifest file to the client device containing URLs that specify the OCAs and files needed for the playback process. Thus, Netflix "controls" the path of the delivery of video content through the network to the Netflix subscribers.



2. A user on a client device requests playback of a title (TV show or movie) from the Netflix application in AWS.
3. The playback application services in AWS check user authorization and licensing, then determine which specific files are required to handle the playback request - taking individual client characteristics and current network conditions into account.



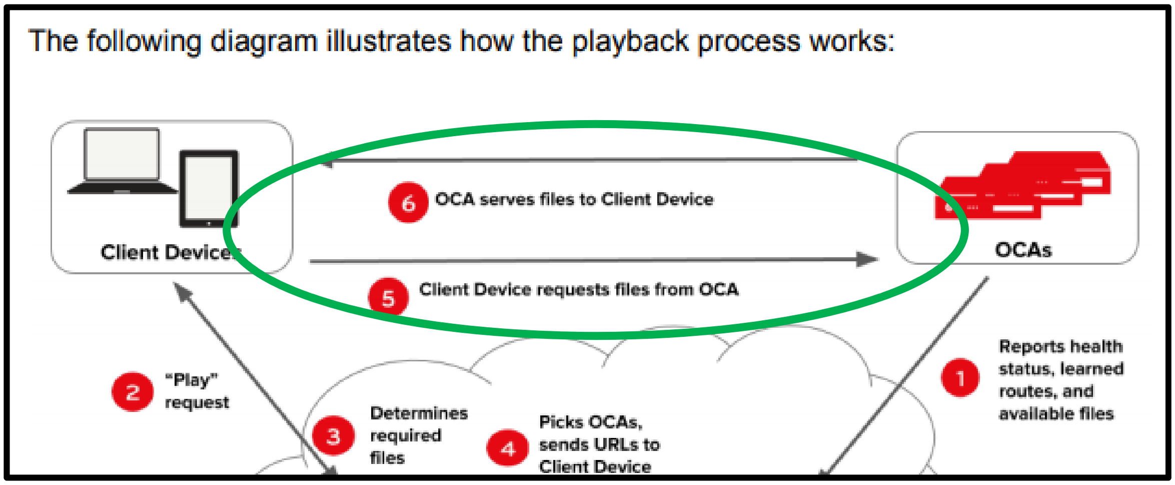
4. The steering service in AWS uses the information stored by the cache control service to pick OCAs that the requested files should be served from, generates URLs for these OCAs, and hands the URLs over to the playback application services.

5. The playback application services hand over URLs of the appropriate OCAs to the client device, and the OCA begins to serve the requested files.

Source: https://openconnect.netflix.com/Open-Connect-Overview.pdf.

77. Through this process, Netflix's content delivery network forms "at least one display pipeline therein." For example, the CDN establishes a link between the client device and one or more Netflix OCAs through which the various video, audio, and other files associated with the requested title are streamed.

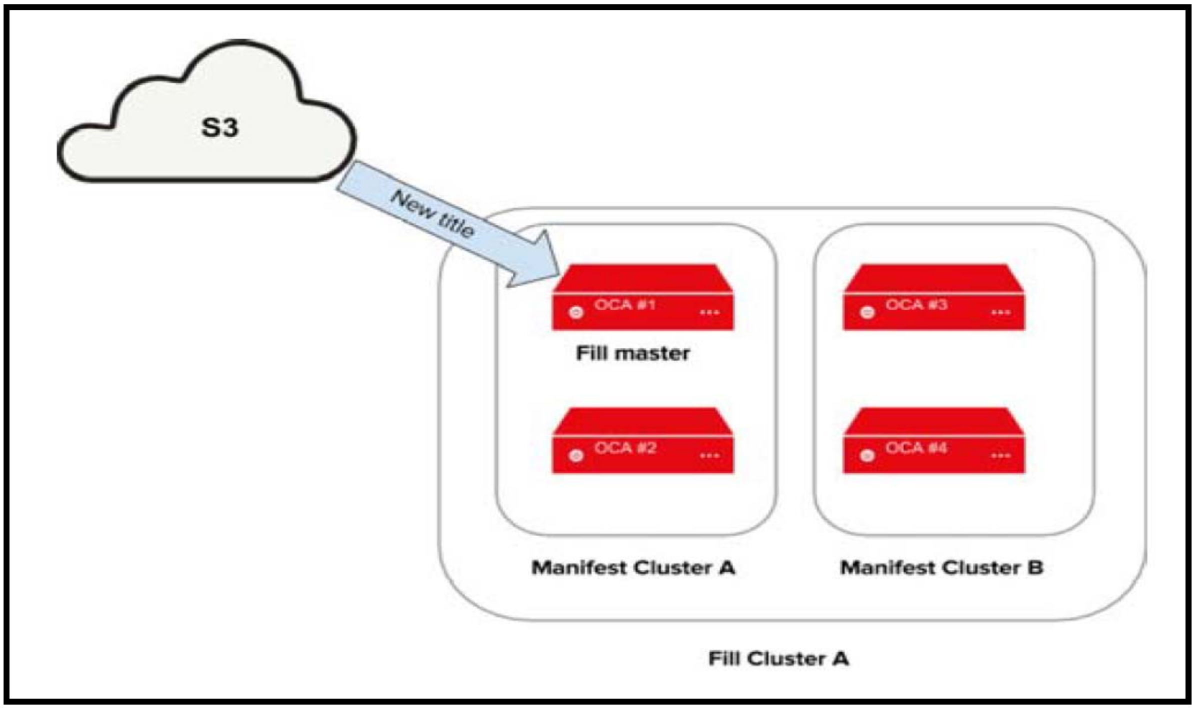
The following diagram illustrates how the playback process works:


Client Device
6 OCA serves files to Client Device
OCAs
5 Client Device requests files from OCA
2 "Play" request
3 Determines required files
4 Picks OCAs, sends URLs to Client Device
1 Reports health status, learned routes, and available files

Source: https://openconnect.netflix.com/Open-Connect-Overview.pdf.

78. The "at least one display pipeline" within Netflix's content delivery network is formed "by dynamically selecting use of at least two selectable nodes from a plurality of selectable nodes." For instance, the content delivery network forms display pipelines between the millions of client devices streaming content from Netflix all over the world ("nodes") and the numerous OCAs and other servers that Netflix uses to stream its content (also "nodes").

79. The OCAs periodically receive new content files—which constitute video and other A/V data—from Netflix's backend systems and from other OCAs.



Source: https://medium.com/netflix-techblog/netflix-and-fill-c43a32b490c0.

The above image, generated by Netflix, shows that the Netflix local OCAs are supplied with content by Netflix's more remotely-located repository of content titles, which creates and stores copies of content that has been transcoded into various formats, as discussed above.

80. Upon receiving a request for content from a client device, the OCA identifies and sends the specific files requested, filtering out the requested data from the numerous content files contained on the OCA.

81. Netflix's selection of nodes is "dynamic." For example, Netflix's "control plane services in AWS take the data that the OCAs report and use it to

steer clients via URL to the most optimal OCAs given their file availability, health, and network proximity to the client."[27]

82. As another example of Netflix's "dynamic" selection of nodes, the Netflix system switches between content files and OCAs during the playback process in order to automatically adapt to network conditions and user behavior. This "dynamic" selection is illustrated, for instance, in Figure 1 of *Characterizing the Workload of a Netflix Streaming Video Server*, a technical paper published in 2016 by the Institute of Electrical and Electronics Engineers ("IEEE"). The portion of this figure within the green box shows, for example, how the Netflix system switches between different video quality levels, or bit rates (shown on the vertical axis), over the course of a Netflix content streaming session (i.e., a Netflix user watching a movie or television program). The horizontal axis shows the elapsed time (in minutes) of that streaming session:

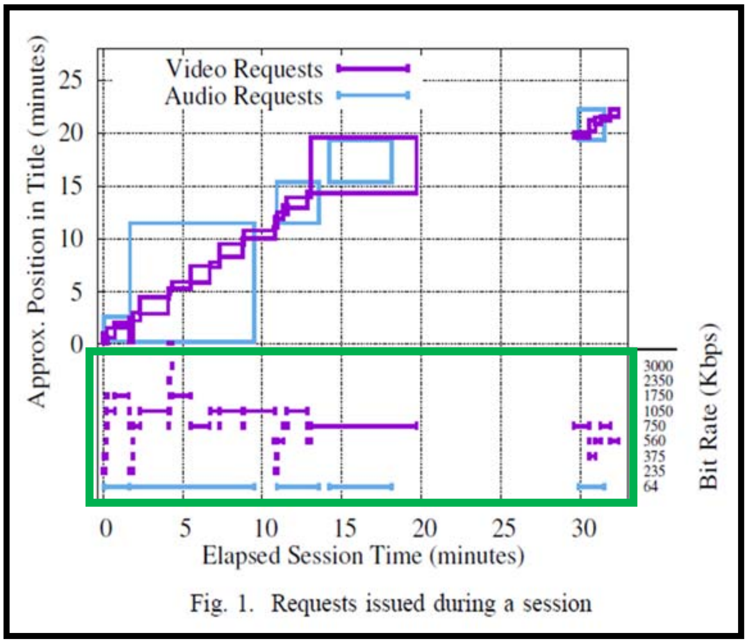
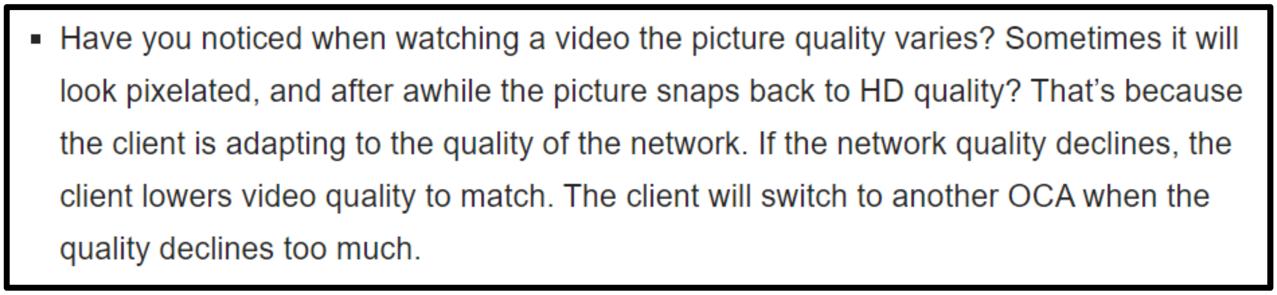


Fig. 1. Requests issued during a session

Source: https://cs.uwaterloo.ca/~brecht/papers/iiswc-netflix-wload-2016.pdf.

[27] https://openconnect.netflix.com/Open-Connect-Overview.pdf.

83. The "dynamic" selection has also been described by other third parties that have analyzed the Netflix system, further illustrating how the Netflix application on the client device—which Netflix controls—automatically switches between URLs for video files and OCAs over the course of the streaming session:

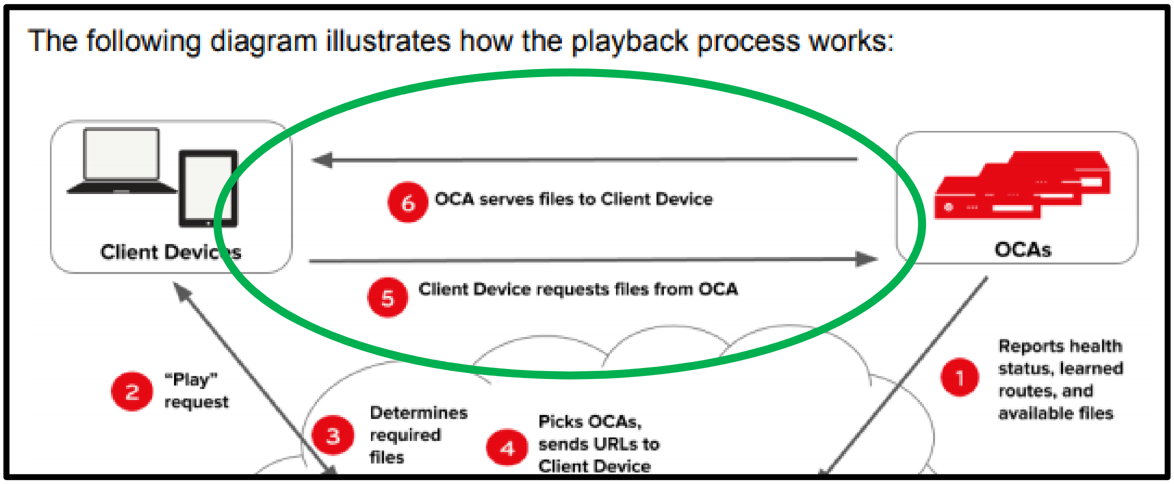


- Have you noticed when watching a video the picture quality varies? Sometimes it will look pixelated, and after awhile the picture snaps back to HD quality? That's because the client is adapting to the quality of the network. If the network quality declines, the client lowers video quality to match. The client will switch to another OCA when the quality declines too much.

Source: http://highscalability.com/blog/2017/12/11/netflix-what-happens-when-you-press-play.html.

84. Thus, the "selection" of the "nodes" is "dynamic" because the node selection can be performed instantaneously, upon the occurrence of network events, and in response to changing network conditions.

85. The Netflix CDN "dynamically concatenate[s] the selected at least two selectable nodes in the network together." The selectable nodes are "concatenated" when a connection is established between them. For example, as explained above, the network establishes a connection between the client device and the best suited OCA through which the client device, using instructions provided by Netflix, "requests files from the OCA" and the "OCA serves files to the Client Device."



The following diagram illustrates how the playback process works:

Client Devices
OCAs
6 OCA serves files to Client Device
5 Client Device requests files from OCA
1 Reports health status, learned routes, and available files
2 "Play" request
3 Determines required files
4 Picks OCAs, sends URLs to Client Device

Source: https://openconnect.netflix.com/Open-Connect-Overview.pdf.

86. As another example of the dynamic nature of the concatenation, the client devices within Netflix's CDN continue to "adapt to the quality of the network…If the network quality declines, the client lowers video quality to match. The client will switch to another OCA when the quality declines too much."[28]

87. The "display pipeline" within the Netflix CDN "has an independent data rate." For example, on information and belief, the data rate of the transmission of streaming content between the OCA and a given client device within the Netflix system varies from and is not dependent upon the data rate of the transmission of streaming content to the other client devices on the system.

88. The Netflix CDN includes a "flow control module" that "enables said independent data rate." For instance, the Netflix client serves as a "flow control module" by "pacing" the transmission of the streaming content from the Netflix OCA. As illustrated below, the client device—under the control of the Netflix application operating thereon—controls the pace at which it downloads content

[28] http://highscalability.com/blog/2017/12/11/netflix-what-happens-when-you-press-play.html.

files from Netflix to address network issues and user events, including by varying the bit rate of the streaming video content.

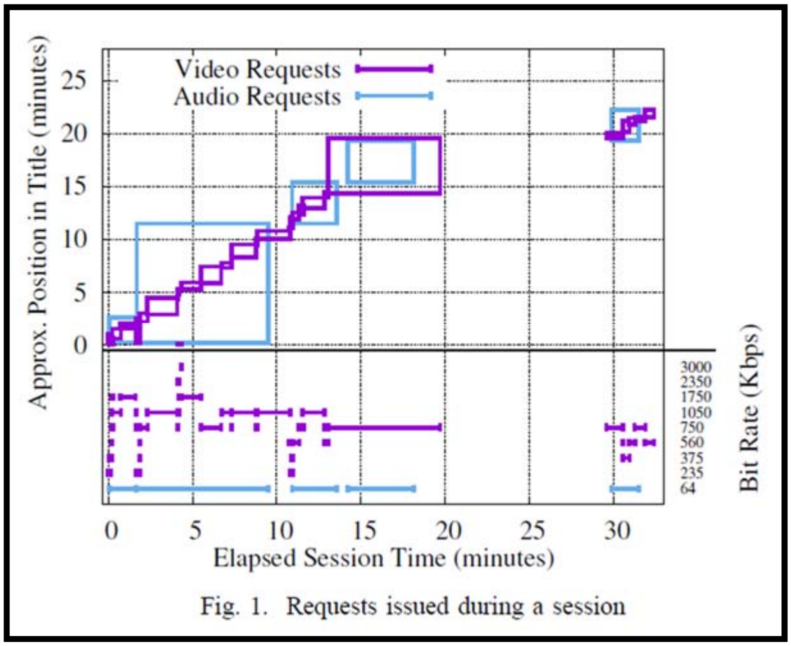
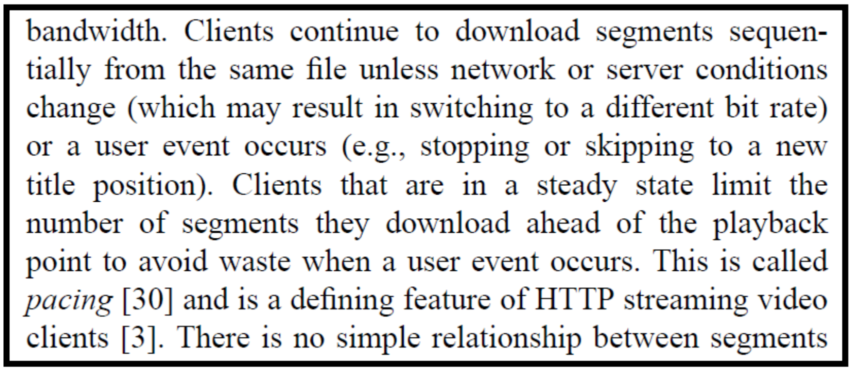


Fig. 1. Requests issued during a session



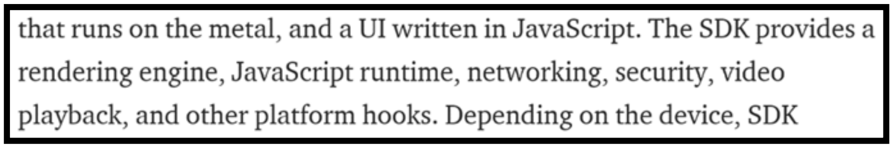
bandwidth. Clients continue to download segments sequentially from the same file unless network or server conditions change (which may result in switching to a different bit rate) or a user event occurs (e.g., stopping or skipping to a new title position). Clients that are in a steady state limit the number of segments they download ahead of the playback point to avoid waste when a user event occurs. This is called *pacing* [30] and is a defining feature of HTTP streaming video clients [3]. There is no simple relationship between segments

Source: https://cs.uwaterloo.ca/~brecht/papers/iiswc-netflix-wload-2016.pdf.

89. Upon information and belief, Netflix controls the playback functionality of the client devices used by subscribers to access the Netflix service. As an example, for many client devices, Netflix develops the software applications used to access the Netflix service. Even in client devices for which Netflix does not develop the application, it "still has control because it controls the software development kit (SDK)." The SDK "is a set of software development tools that allows the creation of applications. Every Netflix app…plays video using the

SDK." "By controlling the SDK, Netflix can adapt consistently and transparently to slow networks, failed OCAs, and any other problems that may arise."[29]

90. In Netflix's words, "[t]he SDK provides a rendering engine, JavaScript runtime, networking, security, *video playback*, and other platform hooks."



that runs on the metal, and a UI written in JavaScript. The SDK provides a rendering engine, JavaScript runtime, networking, security, video playback, and other platform hooks. Depending on the device, SDK

Source: https://medium.com/netflix-techblog/building-the-new-netflix-experience-for-tv-920d71d875de.

91. At least as of on or around September 26, 2019, when the Broadcom Entities informed Netflix of its infringement of the '121 Patent, and by no later than the date of the original complaint in this matter, Netflix has had knowledge of the '121 Patent and that its video streaming service infringes the '121 Patent.

92. In addition to direct infringement, Netflix indirectly infringes the '121 Patent in violation of 35 U.S.C. 271(b) by inducing third-parties to directly infringe at least claim 1 of the '121 Patent, at least in the exemplary manner described above. Netflix has induced, and continues to induce, direct infringement of the '121 Patent by customers and/or end users of infringing client devices enabled with the Netflix software application and service. Netflix knows that it provides and specifically intends to provide an application for use on client devices that, when used as intended with the Netflix streaming service, meets the limitations of claim 1 of the '121 Patent. Netflix knows and specifically intends that its customers and end users practice the system recited in claim 1 of the '121 Patent, when using its

[29] http://highscalability.com/blog/2017/12/11/netflix-what-happens-when-you-press-play.html.

application and service as intended. As explained above, the relevant aspects of that use, including the "playback" process for streaming content, are controlled by Netflix software on the client devices, along with other Netflix systems.

93. Netflix's knowing and willful infringement of the '121 Patent has caused and continues to cause damage to Avago. Avago is entitled to recover damages sustained as a result of Netflix's wrongful acts in an amount subject to proof at trial.

THIRD CLAIM FOR RELIEF

(Infringement of U.S. Patent No. 8,959,245)

94. The Broadcom Entities reallege and incorporate by reference the allegations of paragraphs 1-93 set forth above.

95. The '245 Patent, entitled "Multiple Pathway Session Setup to Support QoS Services" was duly and legally issued on February 17, 2015, from a patent application filed on November 25, 2008, with Jeyhan Karaoguz and James Bennett as the named inventors. A copy of the '245 Patent is attached hereto as **Exhibit C**.

96. The '245 Patent is assigned to Avago, which currently holds all substantial rights, title, and interest in and to the '245 Patent.

97. Pursuant to 35 U.S.C. § 282, the '245 Patent is presumed valid.

98. The '245 Patent is directed to an improvement in the functionality of a communication network, such as the internet.

99. Specifically, the '245 Patent's claims are directed to a novel system and method for delivering content to a user through a communication network, in

which a network management server determines multiple routes for delivering the content based on a provisioning profile for the user device.[30]

100. The inventions of the ’245 Patent resolve technical problems related to delivering content through the then-existing internet network architecture. At the time, the internet was becoming increasingly important as a commercial infrastructure and there was a growing need for “massive internet based services,” such as voice over internet protocol (“VoIP”), video-conferencing, and video streaming. Much of the then-current internet architecture was based on the “best effort” model. This architecture attempts to deliver all data traffic “as soon as possible within the limits of its abilities.” However, the data packets being transferred “can be dropped indiscriminately” in the event of network congestion. While this approach worked well for some less time-sensitive applications, such as email and FTP data transfer, it did not work as well for real-time multimedia applications, such as streaming video on demand. [31]

101. In light of the growing use of high-bandwidth, time-sensitive applications, there was a need for technologies to improve the quality of service (“QoS”) of data transmissions over communication networks like the internet.

102. The inventions described in the ’245 Patent address technical problems associated with the conventional systems and methods for delivering high-quality video, and other data, including by utilizing multiple routes among the available routes in the communication network, thereby increasing the reliability of the data transmission.[32]

[30] ’245 Patent, Abstract, 1:44-51, 2:13-40.
[31] *Id.* at 1:19-40.
[32] *See id.* at 2:29-32, 2:41-45.

103. The ’245 Patent describes and claims specific ways to implement this solution using a network management server capable of determining multiple routes for delivering the data content based on a “provisioning profile,” thereby better ensuring delivery of content over the communication network. The ’245 Patent explains how the provisioning profile may contain information relevant to the delivery of the content, such as preferred service types, desired quality of service, client account information, and/or client credit information.[33]

104. As the ’245 Patent explains, different sets of packets associated with the data content may be transmitted over different routes amongst the multiple available routes to, for instance, take advantage of paths that have less usage and increase reliability. The network management server can manage and prioritize this allocation of routes, which may involve the use of a primary route and/or one or more secondary routes. The network management server can also enable a “handoff” between the routes, such as when QoS degrades on the primary route. The handoff can be seamless to the user, ensuring an uninterrupted user experience.[34]

105. The systems and methods described and claimed in the ’245 Patent improve the functionality of a computer network by providing a new, advantageous approach for delivering content through the network that enables higher QoS standards, such as those required for video-on-demand applications.

106. Claim 1 of the ’245 Patent is directed to:

> 1. A method for communication, the method comprising:

[33] *Id.* at 2:29-32.

[34] *Id.* at 2:41-61.

> receiving from a user device, by a network management server via a communication network, a request for a service;
>
> determining multiple routes for delivering content associated with said requested service based on a provisioning profile for said user device;
>
> and delivering said content associated with said requested service via said determined multiple routes.

107. Claim 1 thus claims a novel solution for transmitting digital media content over a communication network via multiple routes, using a network management server and a provisioning profile. This solution was not well-understood, routine, or conventional at the time of the ’245 Patent because it claims a new and specific improvement over the prior art. The inventions claimed in the ’245 Patent comprise a novel arrangement of streaming content equipment that results in a better experience for the content user with fewer interruptions.

108. Claim 3 of the ’245 Patent, which depends from claim 1, is directed to:

> 3. The method according to claim 1, wherein said provisioning profile comprises preferred service types, desired QoS for one or more services, client account information, and/or client credit verification information.

109. Claim 6 of the ’245 Patent, which also depends from claim 1, is directed to:

> 6. The method according to claim 1, comprising allocating via said network management server, one or more of said determined multiple routes based on priority.

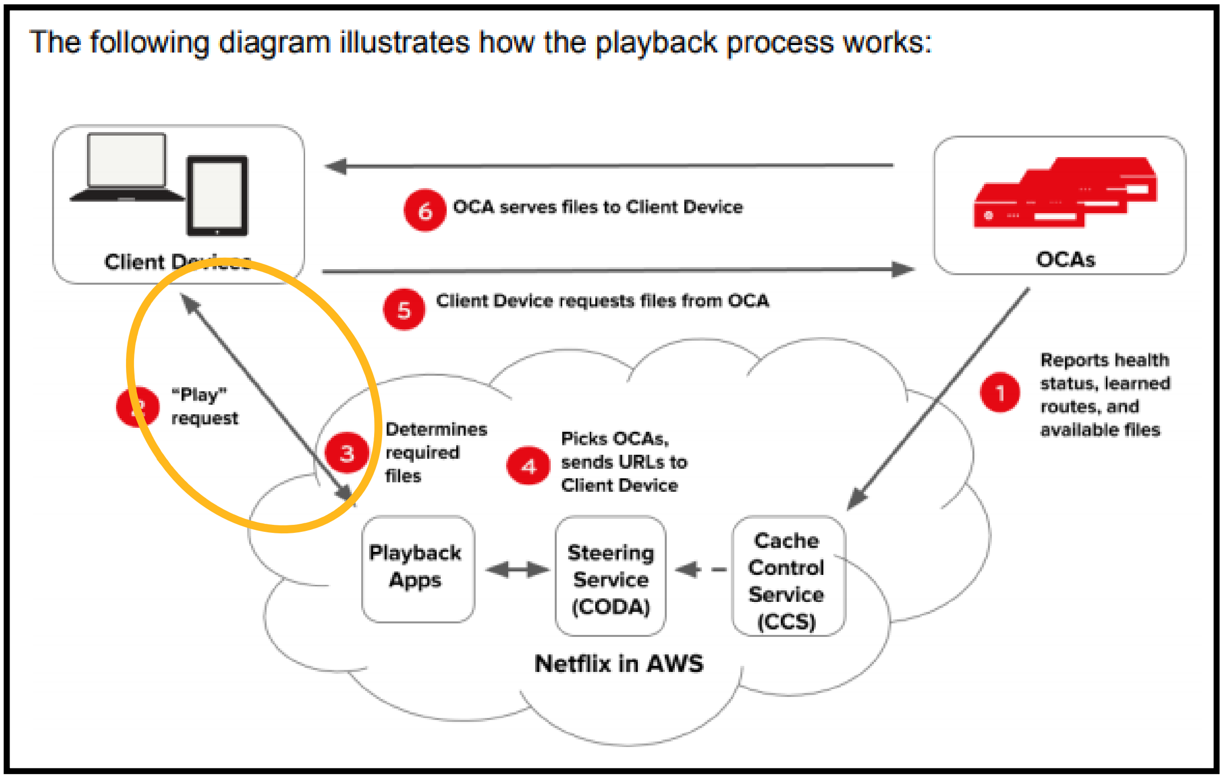
110. Thus, claims 3 and 6 further describe the invention’s improved method whereby a network management server determines multiple routes for delivering content based on a provisioning profile in response to receiving a request for

service from a user device, and then delivers that content via multiple routes based on priority. The ordered combination of elements in each of claims 3 and 6, in conjunction with the elements of the claim from which they depend, therefore recite unconventional, new, and improved digital media content delivery methods that were not well-understood at the time of the ’245 Patent.

111. Netflix directly infringes the ’245 Patent by making, using, offering to sell, and/or selling into the United States its Netflix service, which utilizes a playback system that practices the inventions claimed in the ’245 Patent.

112. Upon information and belief, Netflix directly infringes at least claims 1, 3, and 6 of the ’245 Patent, at least in the exemplary manner described below.

113. Netflix practices a “method for communication” that involves “receiving from a user device, by a network management server via a communication network, a request for a service.” For example, when a Netflix user uses the Netflix application on their TV, computer, smartphone, or other client device to view a movie or television program, the Netflix application sends a “play” request to Netflix’s backend systems. This represents a “request for service” from a “user device.” The Netflix backend systems then determine, using Netflix-developed solutions, the optimal manner in which to deliver the requested content.



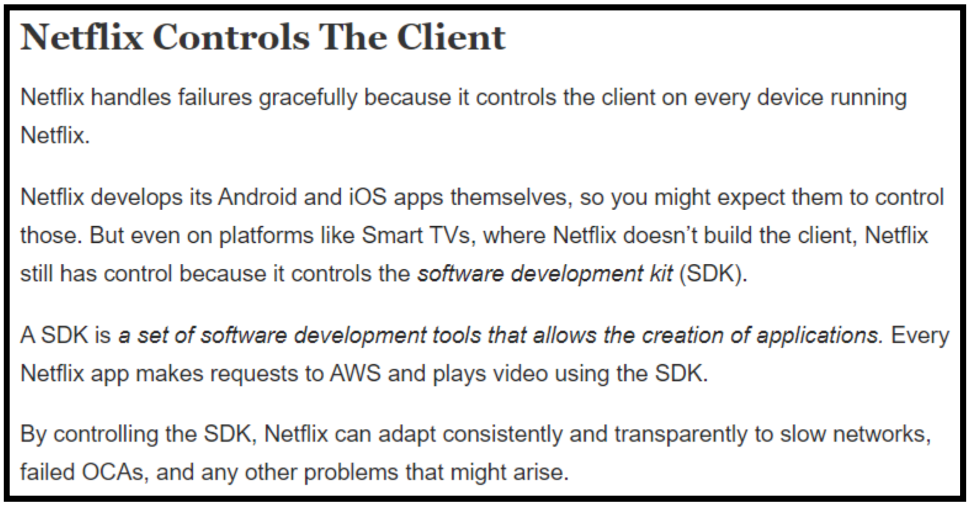
The following diagram illustrates how the playback process works:

Client Devices
OCAs
6 OCA serves files to Client Device
5 Client Device requests files from OCA
1 Reports health status, learned routes, and available files
2 "Play" request
3 Determines required files
4 Picks OCAs, sends URLs to Client Device
Playback Apps
Steering Service (CODA)
Cache Control Service (CCS)
Netflix in AWS

Source: https://openconnect.netflix.com/Open-Connect-Overview.pdf.

114. On information and belief, Netflix controls the playback process from the client side as well. For example, as explained above, Netflix develops many of the Netflix applications used by client devices to access the Netflix service, including the applications for Android and iOS devices. As for the client devices for which Netflix does not develop the application itself, it provides the software development kit, or SDK, used to create the Netflix application. On information and belief, the SDK includes the code responsible for the playback process.

115. Netflix's control of the playback process at the client side is confirmed by third party sources:



Netflix Controls The Client

Netflix handles failures gracefully because it controls the client on every device running Netflix.

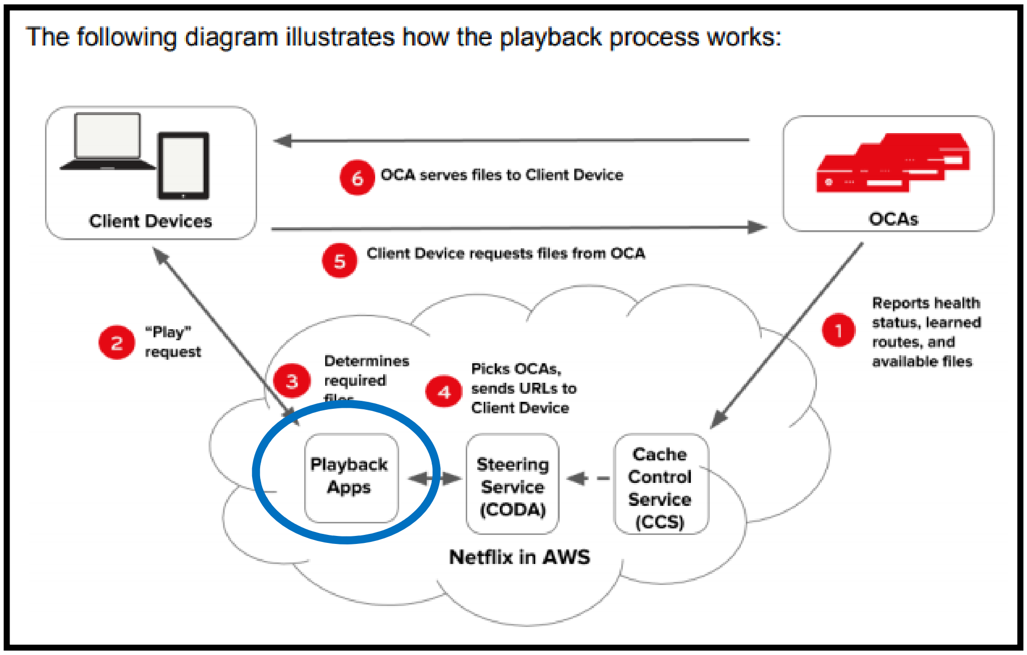
Netflix develops its Android and iOS apps themselves, so you might expect them to control those. But even on platforms like Smart TVs, where Netflix doesn't build the client, Netflix still has control because it controls the *software development kit* (SDK).

A SDK is *a set of software development tools that allows the creation of applications.* Every Netflix app makes requests to AWS and plays video using the SDK.

By controlling the SDK, Netflix can adapt consistently and transparently to slow networks, failed OCAs, and any other problems that might arise.

Source: http://highscalability.com/blog/2017/12/11/netflix-what-happens-when-you-press-play.html.

116. The request for service from the Netflix user's client device is received "by a network management server via a communication network." For example, as shown below, the playback request is received by the Playback Apps service within the Netflix backend, which, on information and belief, operates on one or more servers.



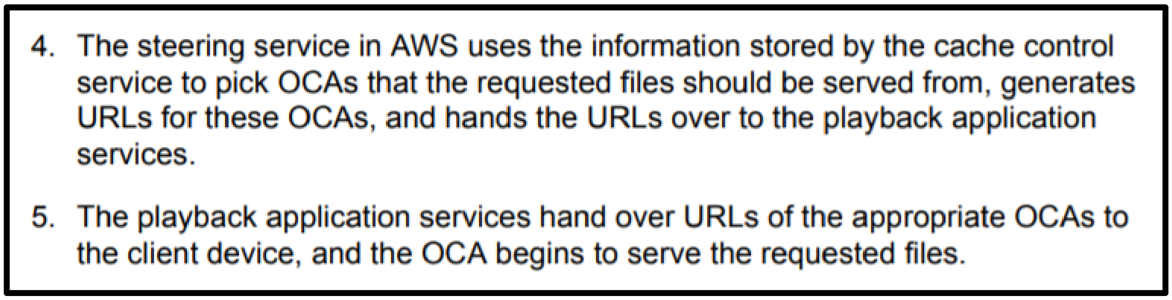
The following diagram illustrates how the playback process works:

Client Devices
6 OCA serves files to Client Device
5 Client Device requests files from OCA
OCAs
1 Reports health status, learned routes, and available files
2 "Play" request
3 Determines required files
4 Picks OCAs, sends URLs to Client Device
Playback Apps
Steering Service (CODA)
Cache Control Service (CCS)
Netflix in AWS

Source: https://openconnect.netflix.com/Open-Connect-Overview.pdf.

117. The Netflix system also "determin[es] multiple routes for delivering content associated with said requested service based on a provisioning profile for said user device." For instance, upon receiving a playback request for a specific title from a Netflix user, the Netflix backend systems determine the video, audio, and other files needed for playback and pick the OCAs from which these files should be streamed to the client device. It then generates a manifest file containing the URLs for the files and OCAs, which the Netflix system sends to the client device.

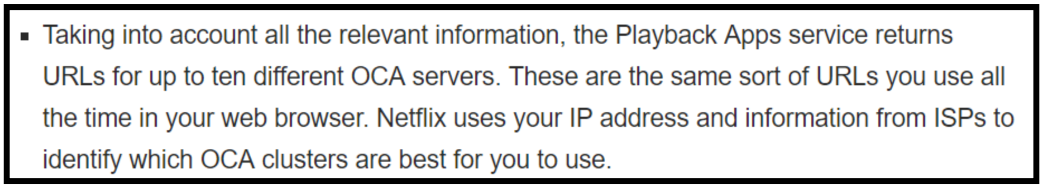
118. As Netflix explains:



4. The steering service in AWS uses the information stored by the cache control service to pick OCAs that the requested files should be served from, generates URLs for these OCAs, and hands the URLs over to the playback application services.
5. The playback application services hand over URLs of the appropriate OCAs to the client device, and the OCA begins to serve the requested files.

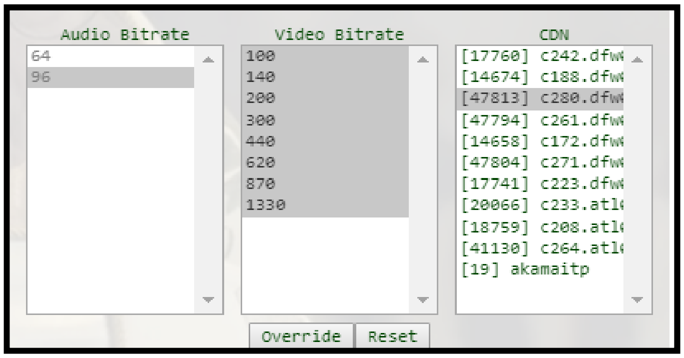
Source: https://openconnect.netflix.com/Open-Connect-Overview.pdf.

119. Through this process, the Netflix system will select as many as ten different OCAs from which the requested content may be streamed:



- Taking into account all the relevant information, the Playback Apps service returns URLs for up to ten different OCA servers. These are the same sort of URLs you use all the time in your web browser. Netflix uses your IP address and information from ISPs to identify which OCA clusters are best for you to use.

Source: http://highscalability.com/blog/2017/12/11/netflix-what-happens-when-you-press-play.html.



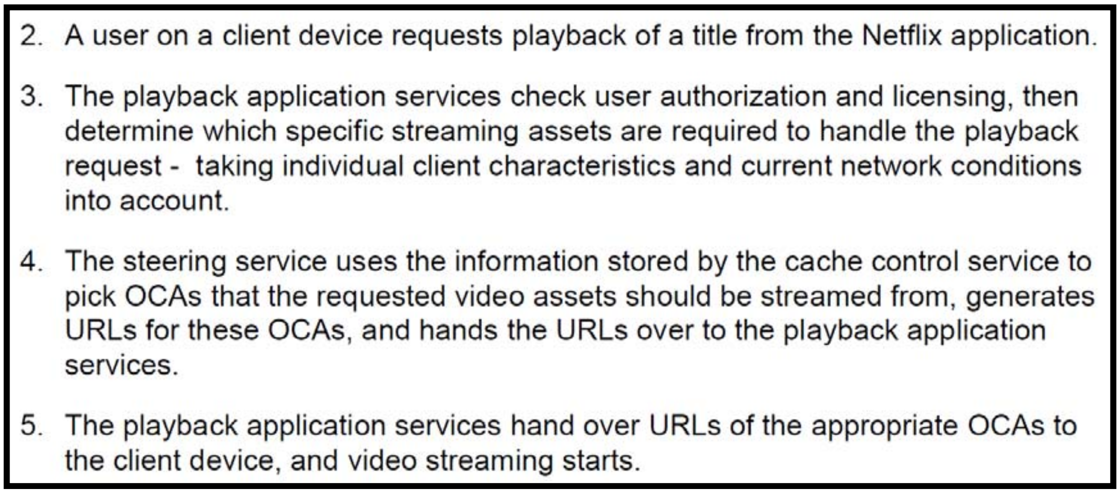
Audio Bitrate
64
96

Video Bitrate
100
140
200
300
440
620
870
1330

CDN
[17760] c242.dfw
[14674] c188.dfw
[47813] c280.dfw
[47794] c261.dfw
[14658] c172.dfw
[47804] c271.dfw
[17741] c223.dfw
[20066] c233.atl
[18759] c208.atl
[41130] c264.atl
[19] akamaitp

Source: Screenshot from Netflix Application.

120. On information and belief, Netflix bases its selection of OCAs and URLs, at least in part, on a "provisioning profile" for the user device used to access the Netflix service. For instance, upon receiving a playback request for a specific title, the Netflix system "checks user authentication and licensing" and takes "individual client characteristics into account" in selecting the "specific streaming assets" required to handle the playback request and the OCAs from which they should be streamed.



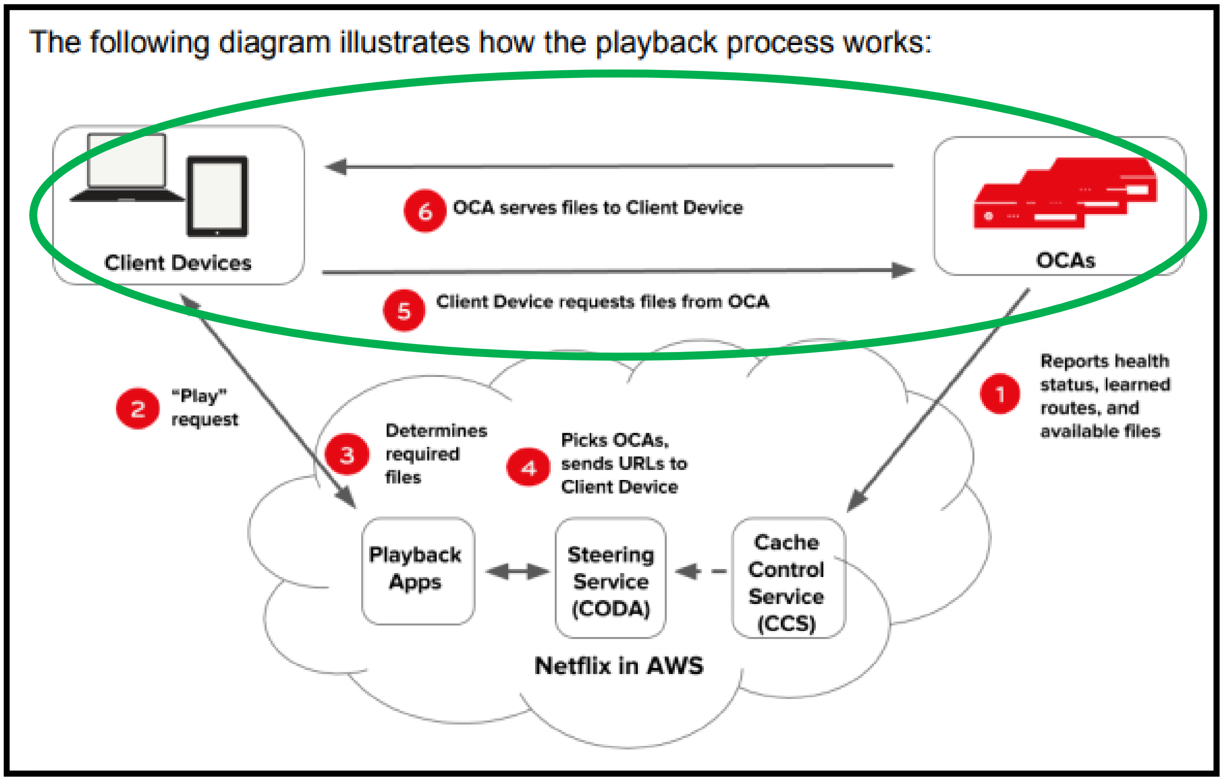
2. A user on a client device requests playback of a title from the Netflix application.
3. The playback application services check user authorization and licensing, then determine which specific streaming assets are required to handle the playback request - taking individual client characteristics and current network conditions into account.
4. The steering service uses the information stored by the cache control service to pick OCAs that the requested video assets should be streamed from, generates URLs for these OCAs, and hands the URLs over to the playback application services.
5. The playback application services hand over URLs of the appropriate OCAs to the client device, and video streaming starts.

Source: https://openconnect.netflix.com/Open-Connect-Overview.pdf.

121. As further explained in a study published by the IEEE, the client device—under the control of the Netflix application—"indicates the formats of the content it can play. Netflix server then sends back a manifest file based upon the

client request. For instance, Netflix client running on an older computer (Thinkpad T60 with Windows XP) and a newer computer (MacBook with Snow Leopard) have different capabilities and received different video downloading format and bit rates."[35]

122. The Netflix system "deliver[s] said content associated with said requested service via said determined multiple routes." For example, the Netflix service connects to one or more of the numerous OCAs within Netflix's CDN and streams video, audio, and other content to the user's computer.


The following diagram illustrates how the playback process works:

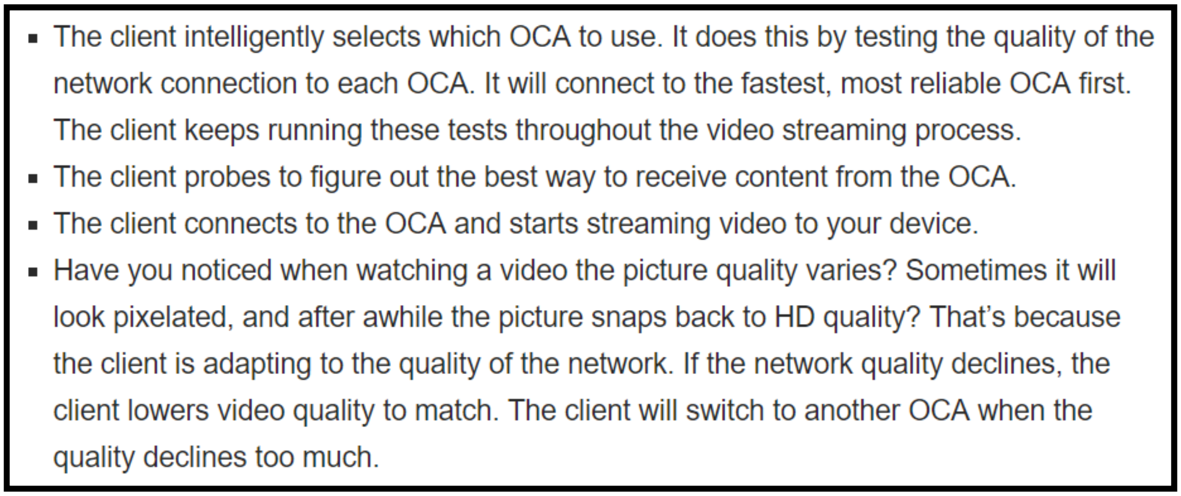
Client Devices
OCAs
6 OCA serves files to Client Device
5 Client Device requests files from OCA
2 "Play" request
1 Reports health status, learned routes, and available files
3 Determines required files
4 Picks OCAs, sends URLs to Client Device
Playback Apps
Steering Service (CODA)
Cache Control Service (CCS)
Netflix in AWS

Source: https://openconnect.netflix.com/Open-Connect-Overview.pdf.

123. In doing so, the Netflix application on the client device intelligently selects which of the specific OCAs identified by Netflix to use by evaluating various factors, including the quality of the network connection to each OCA. The

[35] https://www.moritzsteiner.de/papers/netflix-hulu.pdf.

client device connects to the fastest, most reliable OCA first, but will switch to another OCA if the video quality declines too much.



- The client intelligently selects which OCA to use. It does this by testing the quality of the network connection to each OCA. It will connect to the fastest, most reliable OCA first. The client keeps running these tests throughout the video streaming process.
- The client probes to figure out the best way to receive content from the OCA.
- The client connects to the OCA and starts streaming video to your device.
- Have you noticed when watching a video the picture quality varies? Sometimes it will look pixelated, and after awhile the picture snaps back to HD quality? That's because the client is adapting to the quality of the network. If the network quality declines, the client lowers video quality to match. The client will switch to another OCA when the quality declines too much.

Source: http://highscalability.com/blog/2017/12/11/netflix-what-happens-when-you-press-play.html; https://openconnect.netflix.com/Open-Connect-Overview.pdf.

124. With regard to claim 3 of the '245 Patent, the Netflix system utilizes a "provisioning profile" comprising "preferred service types, desired QoS for one or more services, client account information, and/or client credit verification information." For example, the Netflix system maintains account information for its users. This includes, amongst other thing, preferred playback settings, plan details (e.g., standard or HD), user profile information, parental control information, and credit card payment and billing information. On information and belief, some or all of this account information is used by the Netflix system in selecting the content files and OCAs for use in delivering the title selected by the Netflix user.

125. With regard to claim 6, the Netflix system also allocates multiple routes for delivering the streamed content "based on priority." For instance, on

information and belief, the list of OCAs that Netflix provides to the client device via the manifest file at the start of the playback session is ranked according to priority rules established by Netflix.[36]

126. As alleged above, Netflix directly infringes at least claims 1, 3, and 6 of the '245 Patent.

127. At least as of on or around September 26, 2019, when the Broadcom Entities informed Netflix of its infringement of the '245 Patent, and by no later than the date of the original complaint in this matter, Netflix has had knowledge of the '245 Patent and its infringement thereof.

128. In addition to direct infringement, Netflix has induced, and continues to induce, direct infringement by third-parties of at least claims 1, 3, and 6 of the '245 Patent, at least in the exemplary manner described above, by actively encouraging their use of the Netflix system, in violation 35 U.S.C. § 271(b). Netflix has induced, and continues to induce, direct infringement of the '245 Patent by customers and/or end users of playback devices enabled with the Netflix software application and service. Netflix knows that it provides and specifically intends to provide an application for use on playback devices that, when used as intended with the Netflix streaming service, meets the limitations of claims 1, 3, and 6 of the '245 Patent. Netflix knows and specifically intends that its end users practice the method recited in claims 1, 3, and 6 of the '245 Patent, when using its application and service as intended.

[36] *See*, *e.g.*, http://oc.nflxvideo.net/docs/OpenConnect-Deployment-Guide.pdf; https://www.moritzsteiner.de/papers/netflix-hulu.pdf.

129. Netflix's knowing and willful infringement of the '245 Patent has caused and continues to cause damage to Avago. Avago is entitled to recover damages sustained as a result of Netflix's wrongful acts in an amount subject to proof at trial.

FOURTH CLAIM FOR RELIEF

(Infringement of U.S. Patent No. 8,270,992)

130. The Broadcom Entities reallege and incorporate by reference the allegations of paragraphs 1-129 set forth above.

131. The '992 Patent, entitled "Automatic Quality of Service Based Resource Allocation," was duly and legally issued on September 18, 2012, from a patent application filed on July 18, 2011, with Jeyhan Karaoguz and James D. Bennett as the named inventors. The '992 Patent claims priority to U.S. Provisional Application No. 60/504,876, filed on September 22, 2003. A copy of the '992 Patent is attached hereto as **Exhibit D**.

132. The '992 Patent is assigned to Avago, which holds all substantial rights, title, and interest in and to and '992 Patent.

133. Pursuant to 35 U.S.C. § 282, the '992 Patent is presumed valid.

134. The '992 Patent is directed to an improvement in the functionality of a communication network used to provide a digital media service, such as video streaming over the internet.

135. Specifically, the '992 Patent describes and claims a new system and method for delivering digital media service to users in a dynamic communication network. The invention allocates and utilizes resources from other systems on the network in order to provide the user with the digital media service at a higher

quality level than the quality level they are currently experiencing. As explained in the specification, the resource allocation and utilization can be determined by quality control modules and communication modules. These modules communicate various capability information about the network, such as processing capability, communication capability, and information access capability. The quality control modules can determine whether utilizing resources of another system will provide the service to the user at a higher quality level. If it so determines, a distributing processing module can manage the resource allocation.[37]

136. The inventions of the ’992 Patent address technical problems related to delivering content over unstable network environments, an issue with prior art system existing at the time. In a dynamic and unstable network environment, processing resources continuously toggle between available and unavailable. These processing resources may offer service capabilities that are superior or inferior to other resources present in the network. For example, a system providing low quality audio or video service may communicate with a second system capable of providing a higher quality audio or video service.[38]

137. The first system providing the lower quality service needs a way of both accessing the system with superior resources, utilizing those resources, and ultimately delivering content from the superior resource in order to provide higher quality content to the user.

[37] *See* ’992 Patent, 2:8-32.

[38] *Id.* at 1:39-42 (describing delivery of high quality audio), 1:42-49 (describing delivery of high quality video).

138. The ’992 Patent, therefore, addresses the technical problem of ensuring delivery of content at the highest quality level in an unstable network environment by utilizing a quality-of-service based network resource allocation delivery system.

139. The ’992 Patent claims specific ways to solve these technical problems with a digital media delivery system that is capable of automatically determining bandwidth capability information in multiple systems, using that information to obtain digital media content at a higher quality level than the current quality, and then delivering that higher quality digital media content to the user.[39] In this manner, it improves the functionality of computer networks used to deliver media content by providing a new, advantageous approach for those networks.

140. The digital media delivery system disclosed and claimed in the ’992 Patent uses novel quality control modules, resource allocation modules, and distributing processing modules to assess, manage, and use resources in a network environment where the availability of resources may vary between systems in the network.

141. In a two-system network environment, for example, if it is determined that the second system has access to higher quality audio or video content than the first system, then the first system may receive higher quality data from the second system for further processing. The first system can also receive higher quality data from the system for immediate delivery and presentation to the user. Alternatively, the first system may direct the second system to provide the higher quality data directly to the user.[40]

[39] *Id.* at 3:1-9:16 (describing Figure 1, the method for quality of service based resource allocation and utilization in a dynamic wireless network).
[40] *Id.* at 14:42-53.

142. Claim 1 of the ’992 Patent reads as follows:

> 1. In a portable system, a method for providing a digital media service to a user, the method comprising:
>
> delivering digital media content having a current quality level to a user;
>
> determining that a network connection with a second system is available and is characterized by a communication bandwidth that is high enough to provide the digital media content to the user at a quality level higher than the current quality level;
>
> using the network connection to obtain the digital media content at the higher quality level from the second system; and
>
> delivering the digital media content at the higher quality level to the user instead of the digital media content at the current quality level.

143. Claim 1 thus recites a novel solution of determining, accessing, and using the resources of another system on a dynamic network environment in order to improve digital media content quality delivered to the user by obtaining the higher quality content from the second system in a manner that was not well-understood, routine, or conventional at the time of the ’992 Patent. In the prior art, among other things, the source of the content remained the same between the two systems, whereas claim 1 claims obtaining the digital media content at a higher quality level from another source, namely the second system.

144. Claims 2, 3 and 5 of the ’992 Patent depend from claim 1.

145. Claim 2 of the ’992 Patent reads as follows:

> 2. The method of claim 1, where the digital media content is video media.

146. Claim 3 of the ’992 Patent reads as follows:

> 3. The method of claim 1, where the digital media content is audio media.

147. Claim 5 of the ’992 Patent reads as follows:

> 5. The method of claim 1, where the portable system automatically performs, without user interaction, said determining, said using, and said delivering the digital media content at the higher quality level to the user.

148. Netflix directly infringes the ’992 Patent by making, using, selling, and offering to sell the Netflix service, which practices the patented invention.

149. Upon information and belief, Netflix directly infringes at least claims 1, 2, 3, and 5 of the ’992 Patent, at least in the exemplary manner described below.

150. Netflix practices a “method for providing a digital media service to a user” “[i]n a portable system.”

151. For example, the Netflix system provides a streaming entertainment service that delivers digital video content such as TV series, documentaries, and feature films to a wide variety of internet-connected devices, including mobile devices such as laptops, tablets, and mobile phones.

152. The Netflix system “deliver[s] digital media content having a current quality level to a user.” For example, the Netflix system delivers the “best video quality stream” to its users “tailored to the member’s available bandwidth and view device capability.” To account for variable network conditions, the Netflix streaming service encodes video titles at different bit rates so that video can be delivered at different quality levels. The Netflix streaming service “pre-encode[s] streams at various bitrates applying optimized encoding recipes.”[41]

[41] https://medium.com/netflix-techblog/per-title-encode-optimization-7e99442b62a2.

153. The Netflix system "determin[es] that a network connection with a second system is available and is characterized by a communication bandwidth that is high enough to provide the digital media content to the user at a quality level higher than the current quality level." For instance, the Netflix streaming service is designed to adapt to changing network conditions so that the content can be delivered and viewed at high levels of quality even when network conditions become constrained. Throughout a user's playback session, the Netflix streaming service continuously monitors the network to evaluate changing conditions and makes adjustments to the video that is being delivered.

154. The client device running the Netflix application, which is being directed and controlled by Netflix code and other instructions, "intelligently selects which OCA to use. It does this by testing the quality of the network connection to each OCA. It will connect to the fastest, most reliable OCA first." The client device "keeps running these tests throughout the video streaming process." "If the network quality declines, the client lowers video quality to match. The client will switch to another OCA when the quality declines too much." The Netflix streaming service uses the user's IP address and information from internet Service Providers to adapt to the quality of the network. The Netflix streaming service also identifies which Open Connect Appliance (OCA) clusters are best for the user's client to use. It selects which OCA to use by testing the quality of the network connection to each OCA and will connect to the fastest, most reliable OCA first. This process is repeated throughout the user's video streaming experience.[42]

[42] http://highscalability.com/blog/2017/12/11/netflix-what-happens-when-you-press-play.html.

155. Netflix "us[es] the network connection to obtain the digital media content at the higher quality level from the second system" and "deliver[s] the digital media content at the higher quality level to the user instead of the digital media content at the current quality level." For example, the client device continues to test the quality of the network connection at each OCA throughout the video streaming process. "If the network quality declines, the client lowers video quality to match." The client will switch to another OCA when the quality declines too much. "The Netflix streaming service adapts to the quality of the network and will adjust the video quality by switching to another OCA when the quality declines too much."[43] After establishing a connection to another OCA capable of streaming content at a higher quality level—for example, streaming video at a higher resolution—the Netflix system will provide that content at the higher quality level.

156. With regard to claims 2 and 3, the Netflix system provides "digital media content" in the form of "video media" and "audio media." For example, during the playback process, the Netflix system streams the video and audio files associated with the title being viewed.

157. With regard to claim 5, within the Netflix system, the "portable system automatically performs, without user interaction, said determining, said using, and said delivering the digital media content at the higher quality level to the user." For example, Netflix's process of identifying and using the best OCA and bitrate is generally performed by the Netflix system automatically and without user interaction. As Netflix has explained, the Netflix application on the client device

[43] *Id.*

"runs adaptive streaming algorithms which instantaneously select the best encode to maximize video quality while avoiding playback interruptions due to rebuffers."

158. Netflix has infringed, and continues to infringe, at least claims 1, 2, 3, and 5 of the '992 Patent in the United States by making, using, offering for sale, selling, and/or importing the Netflix streaming service, in violation of 35 U.S.C. § 271(a).

159. At least as of on or around September 26, 2019 when the Broadcom Entities informed Netflix of its infringement of the '992 Patent, and by no later than the date of the original complaint in this matter, Netflix has had knowledge of the '992 Patent and that its video streaming service infringes the '992 Patent.

160. In addition to its own direct infringement, Netflix has induced, and continues to induce, direct infringement by third-parties of at least claims 1, 2, 3, and 5 of the '992 Patent, at least in the exemplary manner described above, by actively encouraging their use of the Netflix system, in violation 35 U.S.C. § 271(b). For example, Netflix has induced, and continues to induce, direct infringement of the '992 Patent by customers and/or end users of infringing playback devices enabled with the Netflix software application and service. Netflix knows that it provides and specifically intends to provide an application for use on playback devices that, when used as intended with the Netflix streaming service, meets the limitations of claims 1, 2, 3, and 5 of the '992 Patent. Netflix knows and specifically intends that its end users practice the method recited in claims 1, 2, 3, and 5 of the '992 Patent, when using its application and service as intended.

161. Netflix's knowing and willful infringement of the '992 Patent has caused and continues to cause damage to Avago. Avago is entitled to recover

damages sustained as a result of Netflix's wrongful acts in an amount subject to proof at trial.

FIFTH CLAIM FOR RELIEF

(Infringement of U.S. Patent No. 6,341,375)

162. The Broadcom Entities reallege and incorporates by reference the allegations of paragraphs 1-161 set forth above.

163. The '375 patent, entitled "Video on demand DVD system" was duly and legally issued on January 22, 2002, from a patent application filed on July 14, 1999, with Daniel Watkins as the named inventor. A copy of the '375 Patent is attached hereto as **Exhibit E**.

164. The '375 Patent is assigned to Broadcom Corp., which holds all substantial rights, title, and interest in and to the '375 Patent.

165. Pursuant to 35 U.S.C. § 282, the '375 Patent is presumed valid.

166. The '375 Patent claims are directed to a new method for distributing video. The '375 Patent describes systems and methods in which the video is delivered to the user on-demand using a drive server, a control server, and one or more decoder devices.[44] The system can process multiple compressed video streams in response to multiple user requests for video content.[45]

167. The '375 Patent includes embodiments described in the specification referring to delivery of compressed video data originating from DVD media. However, the systems and methods are not limited to compressed video from a particular source and, as the patent explains, a person having ordinary skill

[44] '375 Patent at 1:56-60.

[45] *Id*. at 1:60-63.

in the art would recognize that the systems and methods described and claimed in the ’375 Patent can be applied to other video distribution models.[46] The Netflix streaming service is an example of an application of the systems and methods described in the claims.

168. The inventions of the ’375 Patent resolve technical problems related to conventional video-on-demand systems that require the use of physical connections and short distances between the sources of video and the decoders or players on which the end-user views video content.[47] In prior art systems, each user has a dedicated video system, such as a DVD player, and decoder at the user’s location.[48]

169. The ’375 Patent, therefore, addresses the technical problem of ensuring delivery of compressed video content to multiple remote end user locations.[49]

170. The ’375 Patent claims specific ways to solve these technical problems with a video on demand system that is centrally managed and implemented by a drive server, a control server, and one or more decoder devices. Each of these servers can process one or more compressed video streams in response to one or more request signals initiated by a user requesting a video.

171. Claim 15 claims an improved method of distributing video:

> A method for distributing video comprising the steps of:
>
> (A) presenting a plurality of compressed data streams with a drive server to a control server in response to one or more first control signals;
>
> (B) distributing said one or more compressed data streams received from said drive server with said control server to

[46] *Id*. at 5:38-43.
[47] *Id*. at 1:14-41.
[48] *Id*. at 1:26-27, 33-35.
[49] *See id*. at 1:56-2:8.

> one or more decoder devices in response to one or more request signals;
>
> (C) decoding at least one of said one or more compressed data streams with said one or more decoders in response to receiving said one or more compressed data streams from said control server; and
>
> (D) presenting at least one signal selected from a decoded video signal and a decoded audio signal in response to decoding said at least one of said one or more compressed data streams, wherein at least one of said one or more decoders is disposed in a separate room from said control server and said driver server, wherein a first portion of a selected one of said compressed data streams is presented to one of said decoder devices and a second portion of said selected compressed data stream is presented to another of said decoder devices.

172. Claim 15 thus recites a novel solution involving a drive server presenting multiple compressed video streams and delivering those streams to multiple decoder devices in a remote location in a manner that was not well-understood, routine, or conventional at the time of the ’375 Patent, resulting in a better video on demand system.

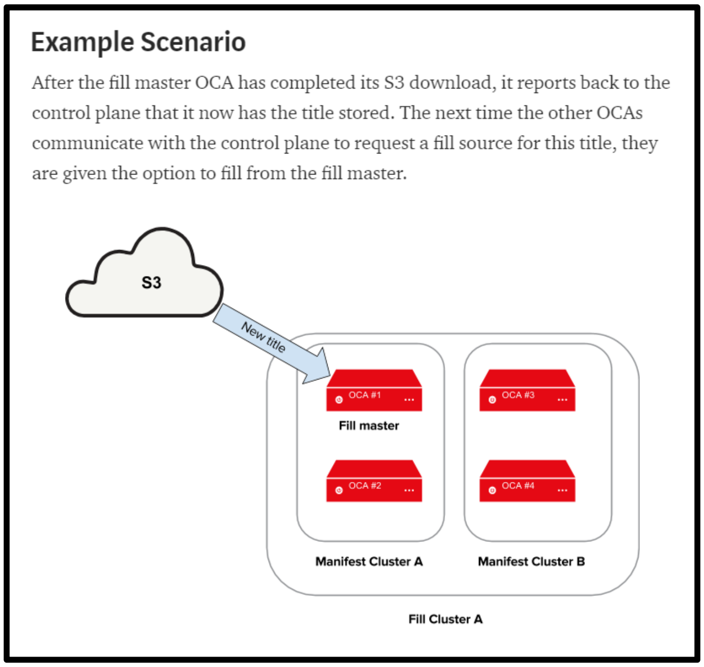
173. Netflix directly infringed the ‘375 Patent by making, using, offering to sell, and/or selling in the United States its Netflix service, which utilizes the inventions claimed in the ‘375 Patent to deliver streaming video content. Although the ’375 Patent is presently expired, Netflix infringed the ’375 Patent prior to its expiration as described below. Broadcom Corp. thus is entitled to damages for Netflix’s unauthorized use of the inventions described in the ’375 Patent prior to its expiration.

174. Upon information and belief, the Netflix streaming service directly infringed at least claim 15 of the '375 Patent, at least in the exemplary manner described below.

175. Netflix practices a "method for distributing video" and "presenting a plurality of compressed data streams with a drive server to a control server in response to one or more first control signals." The Netflix streaming service is a streaming entertainment service that delivers video content using the Netflix CDN, OCAs, and S3 servers. For example, the Netflix video titles ("data streams") are presented from an S3 server or an OCA ("drive server") and sent to other OCAs ("control server"), which store and serve video content.

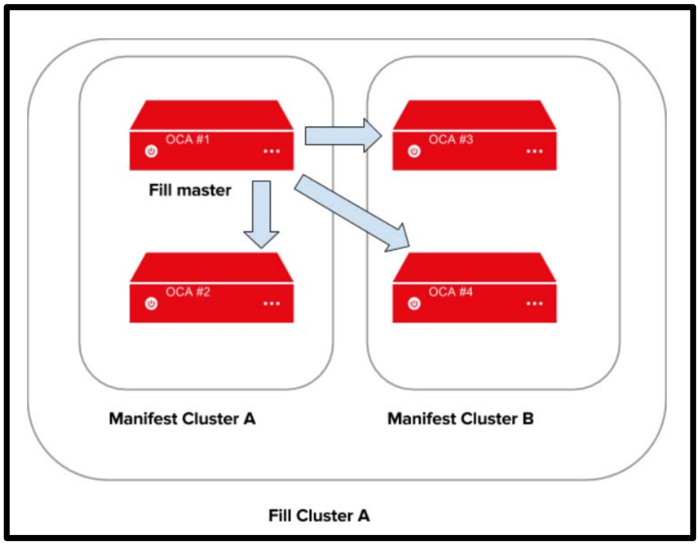
176. The Netflix system presents the compressed data streams to OCAs in response to one or more control signals. For example, as Netflix describes, "OCAs communicate at regular intervals with the control plane services, requesting (among other things) a manifest file that contains the list of titles they should be storing and serving to members. If there is a delta between the list of titles in the manifest file and what they are currently storing, each OCA will send a request, during its configured fill window, that includes a list of the new or update titles that it needs. The response from the control plan in AWS is a ranked list of potential download locations, a.k.a. *fill sources*, for each title."[50]

[50] https://netflixtechblog.com/netflix-and-fill-c43a32b490c0



Example Scenario

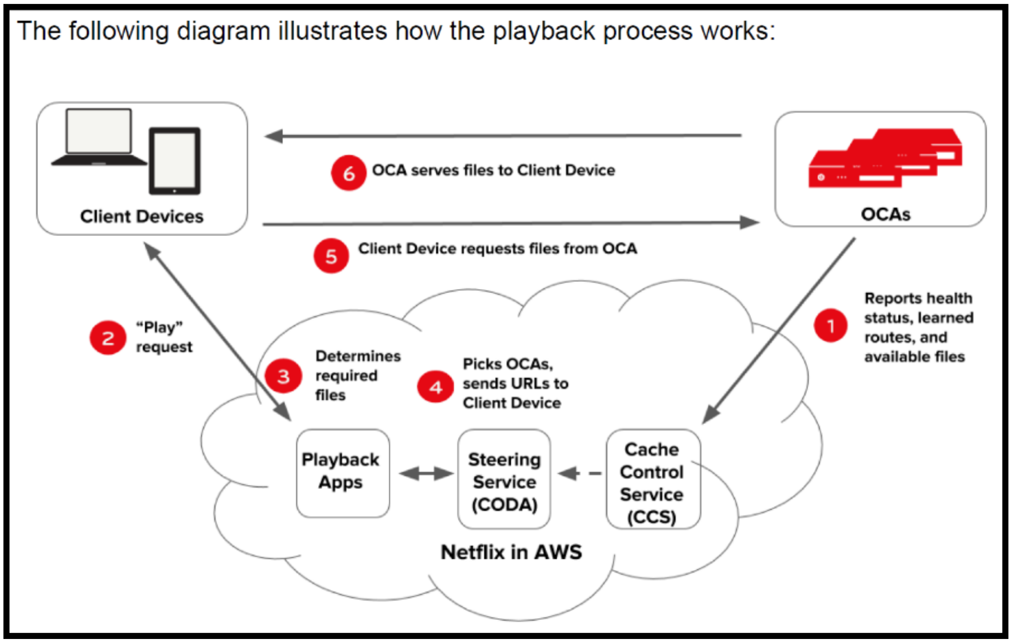
After the fill master OCA has completed its S3 download, it reports back to the control plane that it now has the title stored. The next time the other OCAs communicate with the control plane to request a fill source for this title, they are given the option to fill from the fill master.



Source: https://medium.com/netflix-techblog/netflix-and-fill-c43a32b490c0.

177. Netflix performs the step of "distributing said one or more compressed data streams received from said drive server with said control server to one or more decoder devices in response to one or more request signals." For example, in response to a "Play" request, the Netflix CDN delivers content to Netflix client applications ("Netflix Playback Apps"), which are installed on Netflix's customers' client devices, including digital televisions, desktop computers, laptop computers,

tablet computers, and mobile phones ("decoder devices"). The Netflix CDN is illustrated in the following diagram:

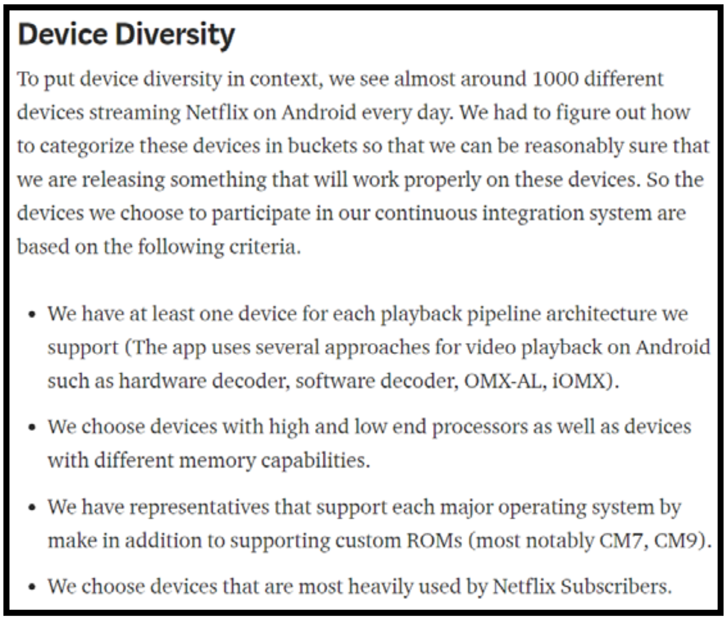


The following diagram illustrates how the playback process works:

Client Devices
OCAs
6 OCA serves files to Client Device
5 Client Device requests files from OCA
2 "Play" request
1 Reports health status, learned routes, and available files
3 Determines required files
4 Picks OCAs, sends URLs to Client Device
Playback Apps
Steering Service (CODA)
Cache Control Service (CCS)
Netflix in AWS

Source: https://openconnect.netflix.com/Open-Connect-Overview.pdf.

178. Netflix practices "decoding at least one of said one or more compressed data streams with said one or more decoders in response to receiving said one or more compressed data streams from said control server." For instance, upon receipt of the compressed video, audio, and other content from the Netflix CDN at the client device, the content is decoded so that it can be viewed. On information and belief, the Netflix Playback Apps decode, or cause the decoding of, the content. For instance, in Netflix's own words, "[t]he app uses several approaches for video playback on Android such as hardware decoder, software decoder, OMX-AL, iOMX").



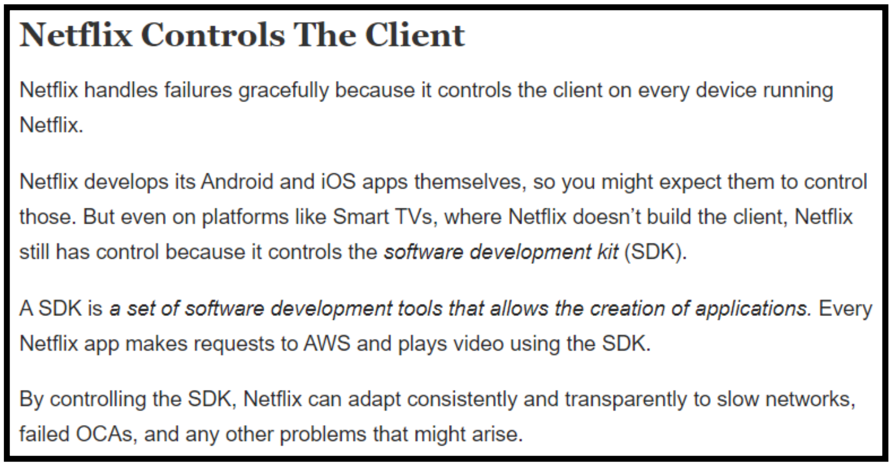
Device Diversity

To put device diversity in context, we see almost around 1000 different devices streaming Netflix on Android every day. We had to figure out how to categorize these devices in buckets so that we can be reasonably sure that we are releasing something that will work properly on these devices. So the devices we choose to participate in our continuous integration system are based on the following criteria.

- We have at least one device for each playback pipeline architecture we support (The app uses several approaches for video playback on Android such as hardware decoder, software decoder, OMX-AL, iOMX).
- We choose devices with high and low end processors as well as devices with different memory capabilities.
- We have representatives that support each major operating system by make in addition to supporting custom ROMs (most notably CM7, CM9).
- We choose devices that are most heavily used by Netflix Subscribers.

Source: http://techblog.netflix.com/2012/03/testing-netflix-on-android.html.

179. On information and belief, Netflix directs and controls the playback process associated with the Netflix service from the client side through its control of the Netflix Playback Apps. Netflix develops its own Android and iOS Netflix Playback Apps for Android and Apple devices. Netflix also develops its own software development kit (SDK) to control third party development of Netflix applications on platforms like Smart TVs.



Netflix Controls The Client

Netflix handles failures gracefully because it controls the client on every device running Netflix.

Netflix develops its Android and iOS apps themselves, so you might expect them to control those. But even on platforms like Smart TVs, where Netflix doesn't build the client, Netflix still has control because it controls the *software development kit* (SDK).

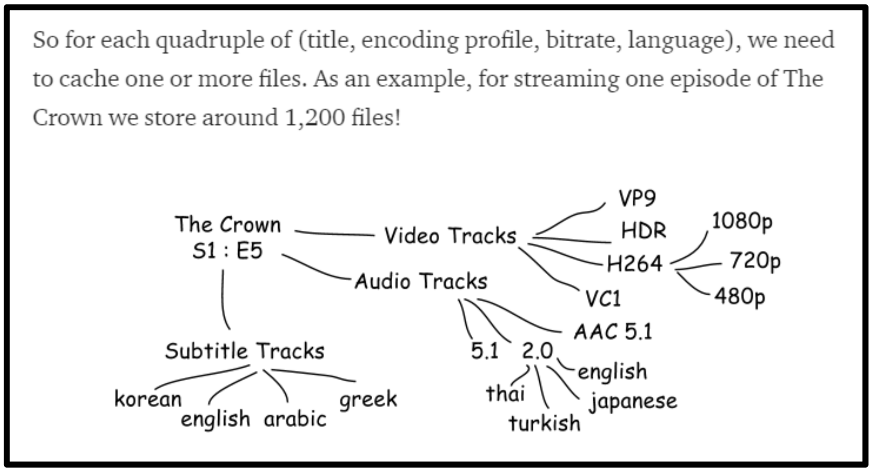
A SDK is *a set of software development tools that allows the creation of applications*. Every Netflix app makes requests to AWS and plays video using the SDK.

By controlling the SDK, Netflix can adapt consistently and transparently to slow networks, failed OCAs, and any other problems that might arise.

Source: http://highscalability.com/blog/2017/12/11/netflix-what-happens-when-you-press-play.html.

180. Netflix practices "presenting at least one signal selected from a decoded video signal and a decoded audio signal in response to decoding said at least one of said one or more compressed data streams, wherein at least one of said one or more decoders is disposed in a separate room from said control server and said driver server, wherein a first portion of a selected one of said compressed data streams is presented to one of said decoder devices and a second portion of said selected compressed data stream is presented to another of said decoder devices."

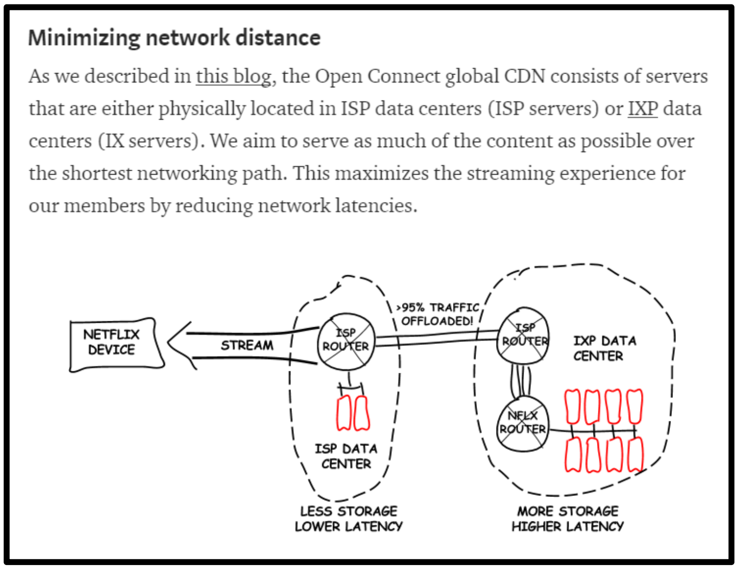
181. For example, a Netflix client—under the control of the Netflix Playback App—presents "at least one signal selected from a decoded video signal and a decoded audio signal in response to decoding said at least one of said one or more compressed data streams" by playing the decoded video, audio, and other Netflix content on the client device. As illustrated below, Netflix content consists of video and audio data, in addition to other data types.



So for each quadruple of (title, encoding profile, bitrate, language), we need to cache one or more files. As an example, for streaming one episode of The Crown we store around 1,200 files!

Source: https://medium.com/netflix-techblog/content-popularity-for-open-connect-b86d56f613b.

182. Further, in the Netflix system, "at least one of said one or more decoders is disposed in a separate room from said control server and said drive server." For example, the Netflix CDN processes video requests and audio requests using OCAs that Netflix strategically positions all over the world based on the location of Netflix's users. The OCAs are located within internet exchange points in significant Netflix markets and are interconnected with internet service providers.[51] Similarly, the S3 is located on one or more servers in data centers that are, generally speaking, far removed from the users streaming Netflix content. In contrast, Netflix users can stream Netflix content virtually anywhere with an internet connection in numerous countries around the world, including the United States. Thus, the "one or more decoders" (in the client devices) "is disposed in a separate room" from the "control server" (the OCA) and the "drive server" (the S3 or OCA).



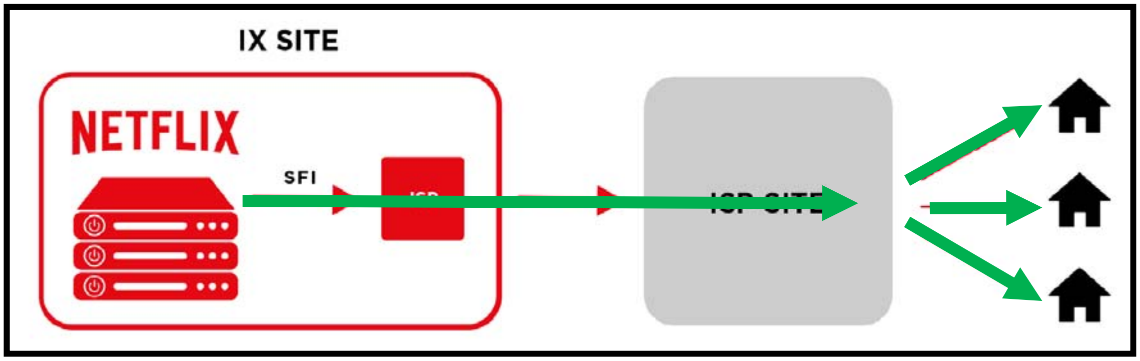
Minimizing network distance

As we described in this blog, the Open Connect global CDN consists of servers that are either physically located in ISP data centers (ISP servers) or IXP data centers (IX servers). We aim to serve as much of the content as possible over the shortest networking path. This maximizes the streaming experience for our members by reducing network latencies.

Source: https://medium.com/netflix-techblog/content-popularity-for-open-connect-b86d56f613b.

[51] https://openconnect.netflix.com/Open-Connect-Overview.pdf.

183. Finally, within the Netflix system, “a first portion of a selected one of said compressed data streams is presented to one of said decoder devices and a second portion of said selected compressed data stream is presented to another of said decoder devices.” For example, on information and belief, when video, audio, and other data associated with Netflix content is streamed from an OCA to the numerous client devices obtaining Netflix content from that OCA, the data travels through the internet infrastructure in a compressed data stream. The relevant portions of that stream are then routed and presented to the appropriate client device.



Source: https://openconnect.netflix.com/Open-Connect-Overview.pdf (arrows added).

184. Netflix directly infringed claim 15, at least as described, when the Netflix Playback Apps, Netflix CDN, and Netflix OCAs were used to stream video content to multiple users in remote and different locations.

185. Netflix’s infringement of the ’375 Patent caused damage to Broadcom Corp., and Broadcom Corp. is entitled to recover damages sustained as a result of Netflix’s wrongful acts in an amount subject to proof at trial.

SIXTH CLAIM FOR RELIEF

(Infringement of U.S. Patent No. 8,572,138)

186. The Broadcom Entities reallege and incorporate by reference the allegations of paragraphs 1-185 set forth above.

187. The ’138 Patent, entitled “Distributed Computing System Having Autonomic Deployment of Virtual Machine Disk Images,” was duly and legally issued on October 29, 2013, from a patent application filed March 30, 2007, with Jagane Sundar, Sanjay Radia, and David A. Henseler as the named inventors. A copy of the ’138 Patent is attached hereto as **Exhibit F**.

188. The ’138 Patent is assigned to Avago, which holds all substantial rights, title, and interest in and to the ’138 Patent.

189. Pursuant to 35 U.S.C. § 282, the ’138 Patent is presumed valid.

190. The ’138 Patent is directed to an improvement in the functionality of a complex distributed computing system. Specifically, the ’138 Patent claims a new method and system for a distributed computing environment that conforms to a multi-level, hierarchical organizational model.

191. Traditional distributed computing systems faced the significant challenge of providing an organizational structure that could handle the deployment and administration of thousands of virtual computing resources that could carry out millions of operations simultaneously. As the ’138 Patent explains, an enterprise environment—such as a large business organization—often includes several business groups, and each group may have competing and variable computing requirements that necessitate separate, independent computing devices connected to each other on the network. However, the diversity of competing and variable

computing requirements increases the cost of distributed computing systems because of the increased time and expense associated with the management of resources that need to be customized to the unique computing requirements of each business group.[52]

192. The ’138 invention solved the technical problems presented by such traditional distributed computing systems by developing an infrastructure management facility (“IMF”) that guarantees reliable and efficient application service delivery independent of the computational infrastructure. The IMF includes the implementation of virtual machine managers, or “container services,” capable of managing other container services and virtual machines (“VMs”). The virtual machines, managed by the VM managers, then appear on the network as available resources as if they were independent computing resources that can be accessed by various groups and utilized to suit their highly-diverse and specialized computing needs.[53]

193. Claim 1 recites an improved method of distributing software “images” (i.e. computer programs) via a number of virtual machines to application nodes:

> A distributed computing system comprising:
>
> a software image repository comprising non-transitory, computer-readable media operable to store: (i) a plurality of image instances of a virtual machine manager that is executable on a plurality of application nodes, wherein when executed on the applications nodes, the image instances of the virtual machine manager provide a plurality of virtual machines, each of the plurality of virtual machine operable to provide an environment that emulates a computer platform, and (ii) a plurality of

[52] ’138 Patent at 1:16-33.
[53] *Id.* at 8:32-36, 9:65-67, 32:65-33:6.

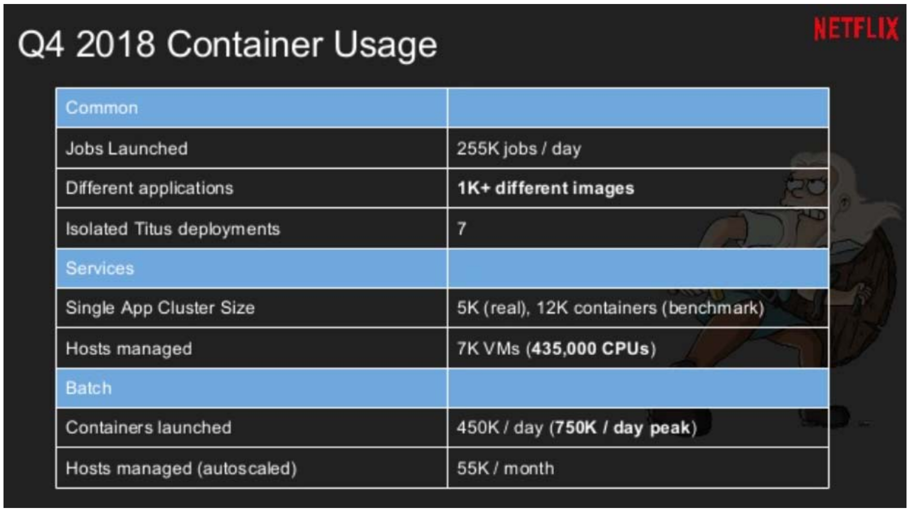
> image instances of a plurality of software applications that are executable on the plurality of virtual machines; and
>
> a control node that comprises an automation infrastructure to provide autonomic deployment of the plurality of image instances of the virtual machine manager on the application nodes by causing the plurality of image instances of the virtual machine manager to be copied from the software image repository to the application nodes and to provide autonomic deployment of the plurality of image instances of the software applications on the virtual machines by causing the plurality of image instances of the software applications to be copied from the software image repository to the application nodes.

194. Netflix directly infringes the ’138 Patent by making, using, offering to sell, and/or selling in the United States its Netflix service, which utilizes Netflix’s Titus Container Management Platform to deploy and manage virtual computing resources in a manner that practices the inventions claimed in the ’138 Patent.

195. Netflix developed the Titus Container Management Platform internally and uses it in production to “power Netflix streaming, recommendations, and content systems.”[54]

196. Using the Titus Container Management Platform, Netflix is able to more efficiently manage the deployment and administration of the hundreds or thousands of VMs necessary for it to provide reliable, high-quality streaming services to its millions of customers. For example, in 2018 Netflix reportedly launched approximately thousands of VM managers and hundreds of thousands of VM containers each day.

[54] https://netflix.github.io/titus/overview/.



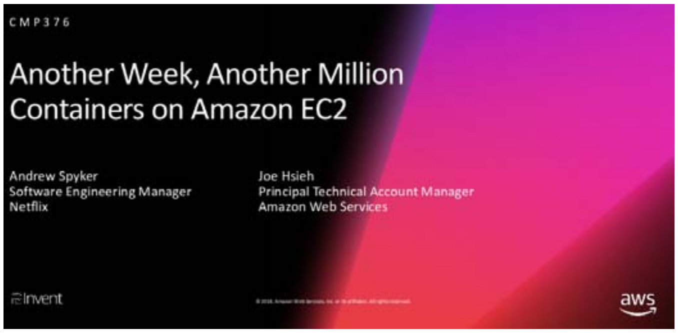
Q4 2018 Container Usage

Common	
Jobs Launched	255K jobs / day
Different applications	**1K+ different images**
Isolated Titus deployments	7
Services	
Single App Cluster Size	5K (real), 12K containers (benchmark)
Hosts managed	7K VMs (**435,000 CPUs**)
Batch	
Containers launched	450K / day (**750K / day peak**)
Hosts managed (autoscaled)	55K / month

Source: https://www.slideshare.net/aspyker/qconsf18-disenchantment-netflix-titus-its-feisty-team-and-daemons?next_slideshow=1.

197. Upon information and belief, Netflix directly infringes at least independent claim 1 of the ’138 Patent at least in the exemplary manner described below.

198. Netflix utilizes a hierarchical “distributed computing system,” as taught by the ’138 Patent, to provide a streaming entertainment service that delivers video content and other content—such as a customer’s browsing experience, content recommendations, and payment information—over a wide geographic area through a distributed network of virtual machines. The Titus architecture is illustrated in the following diagram and description:



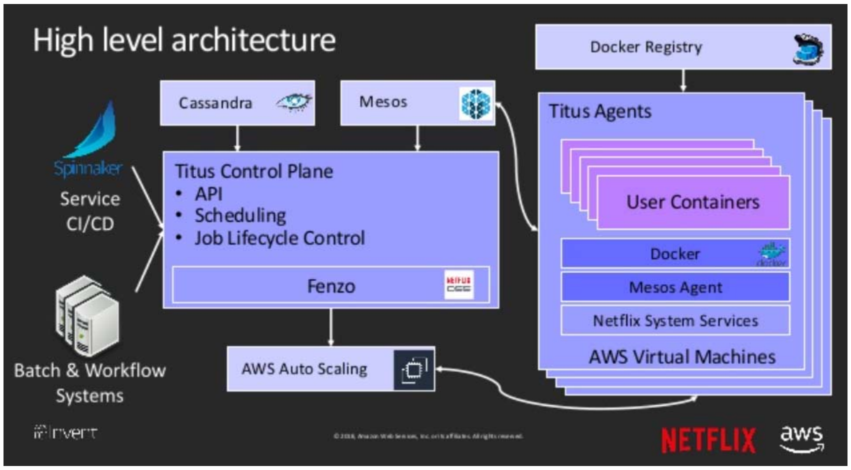
CMP376

Another Week, Another Million Containers on Amazon EC2

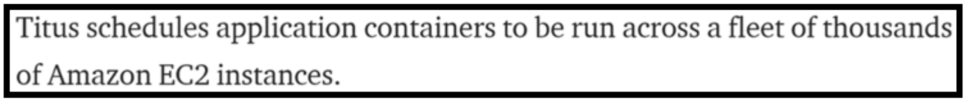
Andrew Spyker
Software Engineering Manager
Netflix

Joe Hsieh
Principal Technical Account Manager
Amazon Web Services

re:Invent
aws



High level architecture



Titus schedules application containers to be run across a fleet of thousands of Amazon EC2 instances.

Source: https://www.slideshare.net/aspyker/cmp376-another-week-another-million-containers-on-amazon-ec2; https://medium.com/netflix-techblog/auto-scaling-production-services-on-titus-1f3cd49f5cd7.

199. The Netflix system uses the Netflix Titus Container Management Platform ("Titus") to manage the distributed computing environment that powers its streaming, recommendation, and content delivery systems.[55] The Titus system integrates a "software image repository," such as Docker Registry, to store various operating system and application files (i.e., "image instances") needed to initialize new instances of virtual machine managers, which Netflix refers to as "Titus Agents."

[55] https://netflix.github.io/titus.

200. The Titus Agent image instances developed by Netflix are executed on the "application nodes"—i.e., "cloud" servers.

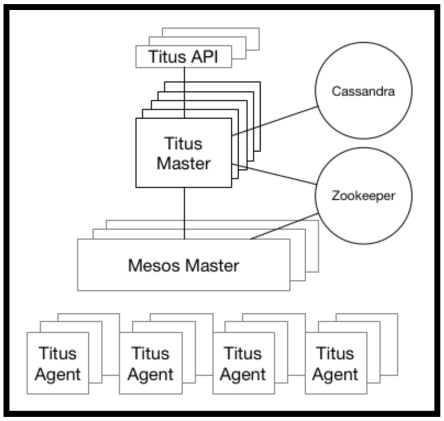
201. For example, as Netflix explains, the Titus Agents set up and manage "virtual machine" containers that are able to independently carry out discrete computing tasks to ensure reliable and efficient delivery of Netflix's streaming services.[56] As in the '138 Patent, each of these virtual machine containers emulates a computer platform and is able to execute various software applications under the supervision of the Titus Agents.

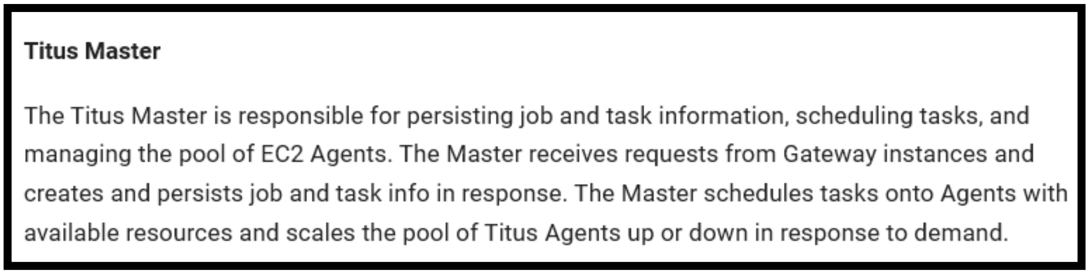
202. The Netflix video distribution system also includes "a control node that comprises an automation infrastructure to provide autonomic deployment of the plurality of image instances of the virtual machine manager on the application nodes by causing the plurality of image instances of the virtual machine manager to be copied from the software image repository to the application nodes and to provide autonomic deployment of the plurality of image instances of the software applications on the virtual machines by causing the plurality of image instances of the software applications to be copied from the software image repository to the application nodes."

203. For example, the Netflix Titus system employs an automation infrastructure known as "Titus Master" to be the "control node" for Netflix's distributed computing system. Like the control node of the '138 invention, the Titus Master provides "autonomic deployment" for new instances of the Titus Agents (i.e., the "virtual machine managers") on the "cloud." The Titus Master is responsible for persisting job and task information, scheduling tasks, and managing

[56] *Id.*

the pool of Titus Agents and can scale the pool of Agents up or down in response to demand.[57] In doing so, the Titus Master causes the appropriate "image instances" (i.e. operating system and software application packages) to be copied from the software image repository to the Titus Agents. The Titus Master is illustrated in the following diagram and description:





Titus Master

The Titus Master is responsible for persisting job and task information, scheduling tasks, and managing the pool of EC2 Agents. The Master receives requests from Gateway instances and creates and persists job and task info in response. The Master schedules tasks onto Agents with available resources and scales the pool of Titus Agents up or down in response to demand.

Source: https://medium.com/netflix-techblog/auto-scaling-production-services-on-titus-1f3cd49f5cd7; https://netflix.github.io/titus/overview/.

204. At least as of on or around September 26, 2019, when the Broadcom Entities provided Netflix with an exemplary infringement chart for the '138 Patent, and by no later than the date of the original complaint in this matter, Netflix has had knowledge of the '138 Patent and its infringement thereof.

[57] https://medium.com/netflix-techblog/the-evolutionof-container-usage-at-netflix-3abfc096781b; https://medium.com/netflix-techblog/titusthe-netflix-container-management-platformis-now-open-source-f868c9fb5436.

205. Netflix's knowing and willful infringement of the '138 Patent has caused and continues to cause damage to Avago. Avago is entitled to recover damages sustained as a result of Netflix's wrongful acts in an amount subject to proof at trial.

SEVENTH CLAIM FOR RELIEF

(Infringement of U.S. Patent No. 6,744,387)

206. The Broadcom Entities reallege and incorporate by reference the allegations of paragraphs 1-205 set forth above.

207. The '387 Patent, entitled "Method and System for Symbol Binarization," was duly and legally issued on June 1, 2004 from a patent application filed on July 10, 2002, with Lowell Winger as the named inventor. A copy of the '387 Patent is attached hereto as **Exhibit G**.

208. The '387 Patent is assigned to Broadcom Corp., which holds all substantial rights, title, and interest in and to the '387 Patent.

209. Pursuant to 35 U.S.C. § 282, the '387 Patent is presumed valid.

210. The '387 Patent generally concerns an improvement in the way a computer system compresses visual and audio data. Specifically, the patent is "directed to an improved method for the binarization of data in an MPEG data stream."[58]

211. As the patent explains, MPEG refers to a family of international standards developed by the Motion Picture Expert Group that specify how to represent visual and audio information in a compressed digital format.[59] The

[58] '387 Patent, Abstract; *see also id.* at 1:7-11.
[59] *Id.* at 1:15-30.

MPEG formats make it possible to "represent[] a video signal with data roughly 1/50th the size of the original uncompressed video, while still maintaining good visual quality."[60] The MPEG formats achieve such high compression by taking advantage of the fact that many images in a video stream do not change significantly from picture to picture, and if they do change, the differences from one picture to the next are often simple.[61] Storing and transmitting only the changes, instead of entire pictures, results in considerable savings in data transmission.[62]

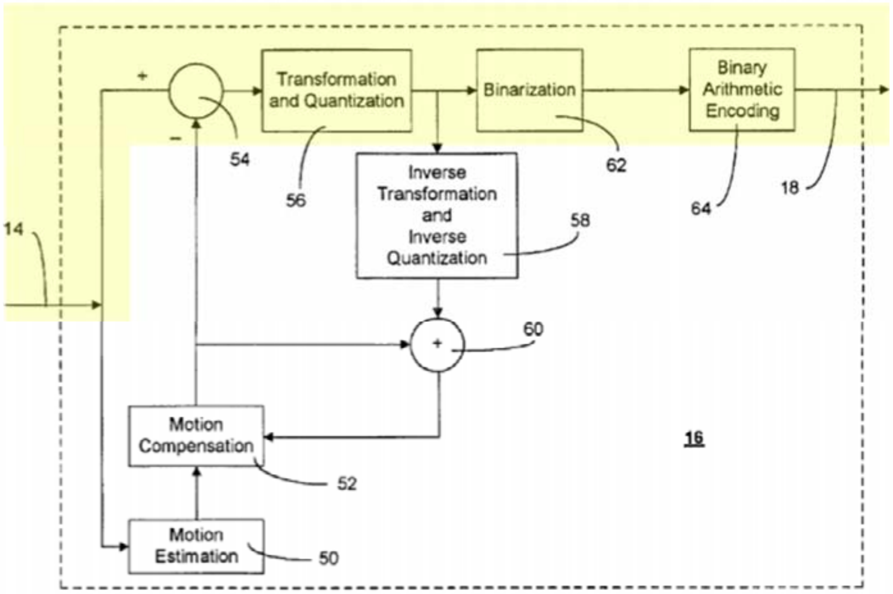
212. In practice, this compression technique is accomplished in a number of steps. First, pixel differences between the pictures are "transformed into frequency coefficients, and then quantized to further reduce the data transmission."[63] The '387 Patent refers to the resulting transformed and quantized coefficients as "symbols."[64] Second, the transformed-quantized symbols are "binarized" to create binary representations of each symbol in the form of a "codeword."[65] Third, the binarized codewords are entropy encoded "to reduce the number of bits per symbol without introducing any additional video signal distortion."[66] The patent explains that several types of codecs[67] exist for performing the entropy encoding; "[o]ne of the most efficient of which is the family of binary arithmetic encoders (BACs)."[68]

[60] *Id.* at 1:36-39.
[61] *Id.* at 3:29-37.
[62] *Id.* at 1:51-54, 3:34-37.
[63] *Id.* at 1:55-58, 3:40-43.
[64] *See*, *e.g.*, *id.* at 1:55-64, 4:1-4, 4:45-54, 5:10-46, Table 1.
[65] *Id.* at 4:1-4.
[66] *Id.* at 1:59-63, 4:26-36.
[67] A codec is a device or computer program for encoding or decoding a digital stream or signal.
[68] '387 Patent, 4:37-39.

As the name implies, BACs operate only on binary valued data, which is why the symbols must be binarized before they can be entropy encoded.[69]

213. Figure 2 of the ’387 Patent—the relevant portion of which is highlighted below—is a block diagram of an encoder that carries out this encoding process.

FIG. 2


Transformation and Quantization
Binarization
Binary Arithmetic Encoding
Inverse Transformation and Inverse Quantization
Motion Compensation
Motion Estimation
+
−
+
54
56
62
64
18
14
58
60
52
50
16

Source: ’387 Patent, Fig. 2 (highlighting added).

214. Figure 2 depicts that a source video stream (14) enters the encoder (16) at the left and passes to a “combination” module (54), which assembles data related to pixel differences between pictures in the video stream.[70] The output of the combination module passes to the next module (56), where it is transformed and quantized.[71] Symbols created by module 56 pass to the binarization module (62),

[69] *Id.* at 4:45-49.
[70] *Id.* at 3:49-53.
[71] *Id.* at 3:54-57.

which creates binary codewords that represent the symbols. The codewords next pass to binary arithmetic encoding module (64), where they are entropy encoded. And finally, the encoded bitstream (18) exits the encoder at the right.[72]

215. The ’387 Patent is directed to the step in this process that occurs in the binarization module 62.[73] In the prior art, several techniques were available for binarizing the symbols, including, for example: unary, binary, Golomb, and exp-Golomb binarization.[74] The patent explains that those techniques each have certain strengths and weaknesses. Unary binarization, for example, generates codewords that are more easily distinguishable from one another but that can be exceptionally long. Specifically, “[u]nary binarization consists of a number of binary 1s equal to an index for a symbol followed by a zero….”[75] Thus, a symbol index value of “1” results in a 2-bit codeword, namely “10.” A symbol index value of “2” results in the 3-bit codeword “110,” and so on. Thus, each codeword is easily distinguishable from the others, but the number of binary values can be quite large—encoding a large symbol index may require tens of thousands of bits.[76]

216. Exp-Golomb binarization, on the other hand, greatly reduces the maximum possible size of the codewords, but “it does not permit codewords with a small symbol index (other than index 0) to be uniquely distinguished from codewords with larger symbol indices.”[77]

[72] *Id.* at 3:57-4:8.
[73] *Id.* at 4:1-2.
[74] *Id.* at 4:50-51.
[75] *Id.* at 4:52-54.
[76] *Id.* at 5:22-5:45.
[77] *Id.* at 6:19-24.

217. Thus, there was a need for a method and system that could exploit the most valuable properties of the unary and exp-Golomb binarizations.[78] The invention described in the ’387 Patent meets that need:

> The present invention provides a binarization that retains the most valuable properties of the unary and exp-Golomb binarizations. That is, small codewords are distinguishable as with a unary binarization, while large codewords have their binarization limited to a reasonable length. By doing so, the present invention provides a binarization that reduces the complexity and the bitrate/size for compressing and decompressing video, images, and signals that are compressed using binary arithmetic encoding for entropy encoding.[79]

218. The ’387 Patent claims methods and systems for constructing binarized codewords for digital video data (i.e., encoding) based on the index values of symbols produced by the transformation-and-quantization module of an encoder. Independent claim 3, for example, recites:

> A binarization system comprising:
>
> means for determining if a code symbol index value is less than a threshold value
>
> means for constructing a codeword using a unary binarization if said code symbol index value is less than said threshold value; and
>
> means for constructing a codeword using a exp-Golomb binarization if said code symbol index value is not less than said threshold value.

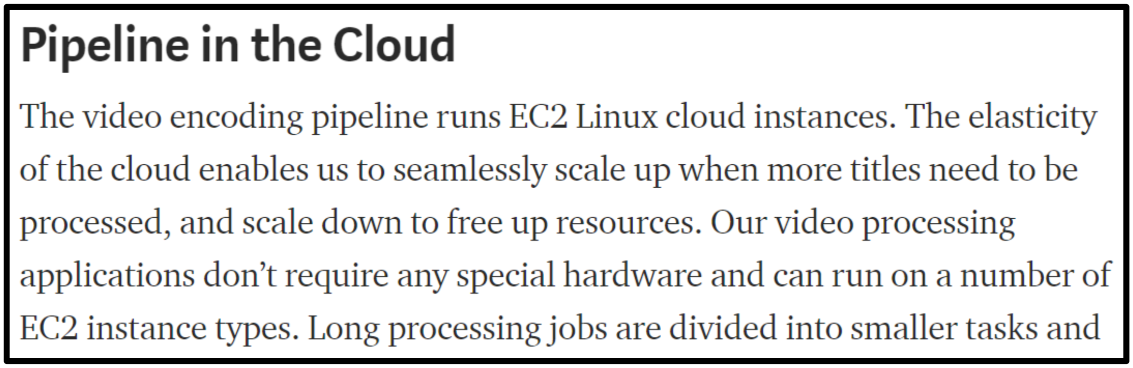
219. The methods and systems described in the ’387 Patent improve the functionality of computer systems by improving the way they compress and process video and audio data.

[78] *Id.* at 2:2-12.
[79] *Id.* at 6:26-36.

220. Notably, the Hon. James V. Selna of this District previously held that the claims of the ’387 Patent are patent-eligible under 35 U.S.C. § 101. In doing so, Judge Selna concluded that these claims “do not simply use general computers to perform abstract ideas; instead, the mathematical formula attempts to improve the functioning of compressing and decompressing video, images, and signals. Therefore, an inventive concept sufficiently transforms the nature of the claims…into patent-eligible inventions.”[80]

221. Netflix directly infringes at least claim 3 of the ’387 Patent at least in the exemplary manner described below.

222. Netflix developed, operates, and uses a “video encoding pipeline”, i.e., a series of video processing applications that operate “in the cloud.” Netflix claims that it has developed, and that it uses, its own proprietary video encoding software in this pipeline.



Pipeline in the Cloud

The video encoding pipeline runs EC2 Linux cloud instances. The elasticity of the cloud enables us to seamlessly scale up when more titles need to be processed, and scale down to free up resources. Our video processing applications don’t require any special hardware and can run on a number of EC2 instance types. Long processing jobs are divided into smaller tasks and

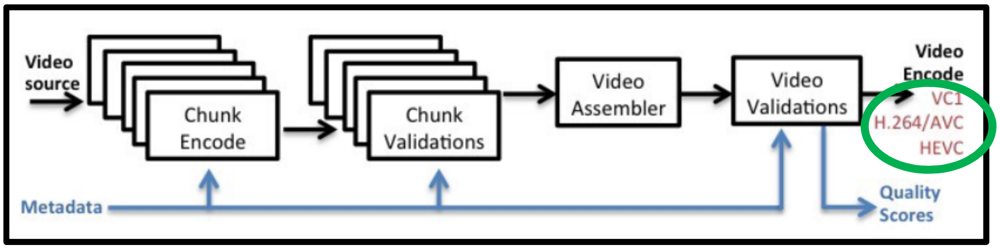
Source: https://medium.com/netflix-techblog/high-quality-video-encoding-at-scale-d159db052746.

[80] *Broadcom Corp., et al. v. Sony Corp., et al.*, SAVC 16-1052 JVS (JCGx), p. 12 (C.D.C.A. Oct. 5, 2016).

223. Netflix uses its video encoding pipeline to generate encoded video files in a variety of formats, which it then uses to stream movie and TV content to its subscribers. As Netflix explains:

> We ingest high quality video sources and generate video encodes of various codec profiles, at multiple quality representations per profile. The encodes are packaged and then deployed to a content delivery network for streaming. During a streaming session, the client requests the encodes it can play and adaptively switches among quality levels based on network conditions.[81]

224. Among other encodes, Netflix uses its video encoding pipeline to generate content files in the ITU-T H.264 format, also known as Advanced Video Coding or "AVC" ("H.264"), and in the ITU-T H.265 format, also known as High Efficiency Video Coding or "HEVC" ("H.265").



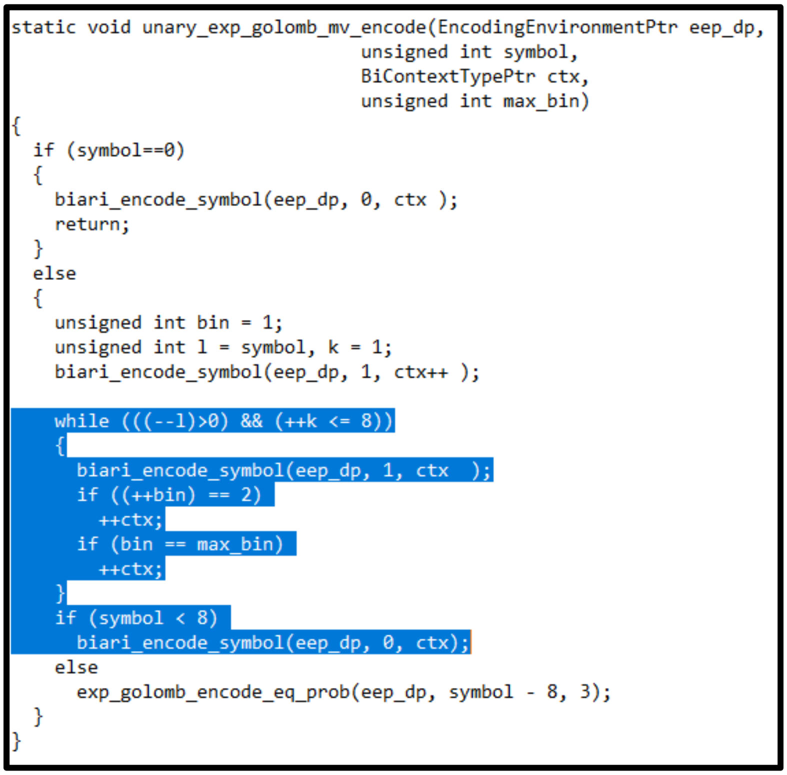
Source: https://medium.com/netflix-techblog/high-quality-video-encoding-at-scale-d159db052746.

225. The Netflix video encoding pipeline includes a "binarization system." For example, with regard to the H.264 format used by Netflix, the H.264 documentation explains how the format employs a "concatenated unary/k-th order Exp-Golomb (UEGk) binarization process." On information and belief, Netflix

[81] https://medium.com/netflix-techblog/high-quality-video-encoding-at-scale-d159db052746.

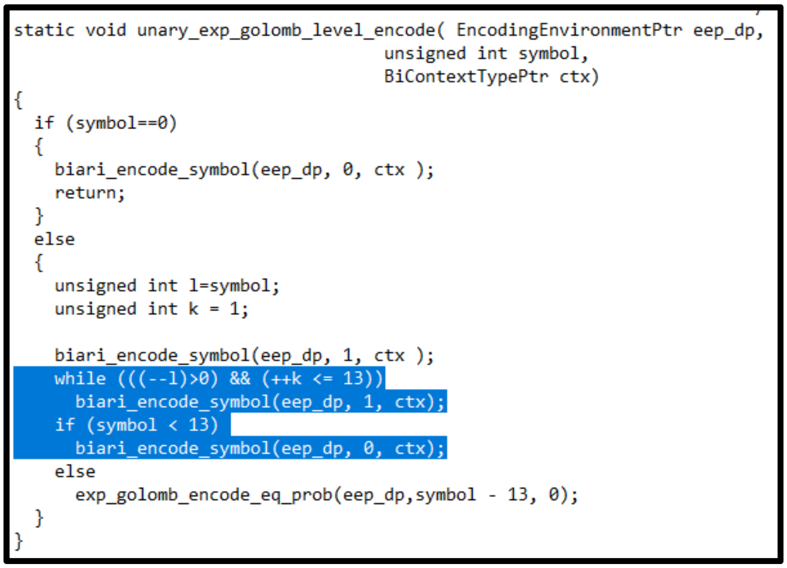
uses the method for UEG(k) encoding set forth in the H.264.2 reference software, which serves as an aid for the study and implementation of H.264 video coding.

226. On information and belief, the Neflix video encoding pipeline includes a "means for determining if a code symbol index value is less than a threshold value." For instance, the H.264.2 reference software features the functions unary_exp_golomb_mv_encode() and unary_exp_golomb_level_encode(), which are responsible for performing unary exp-golomb encoding for various syntax elements. Both of these functions employ code that determines if a symbol index value is less than a threshold value.



```
static void unary_exp_golomb_mv_encode(EncodingEnvironmentPtr eep_dp,
                                        unsigned int symbol,
                                        BiContextTypePtr ctx,
                                        unsigned int max_bin)
{
  if (symbol==0)
  {
    biari_encode_symbol(eep_dp, 0, ctx );
    return;
  }
  else
  {
    unsigned int bin = 1;
    unsigned int l = symbol, k = 1;
    biari_encode_symbol(eep_dp, 1, ctx++ );

    while (((--l)>0) && (++k <= 8))
    {
      biari_encode_symbol(eep_dp, 1, ctx  );
      if ((++bin) == 2)
        ++ctx;
      if (bin == max_bin)
        ++ctx;
    }
    if (symbol < 8)
      biari_encode_symbol(eep_dp, 0, ctx);
    else
      exp_golomb_encode_eq_prob(eep_dp, symbol - 8, 3);
  }
}
```

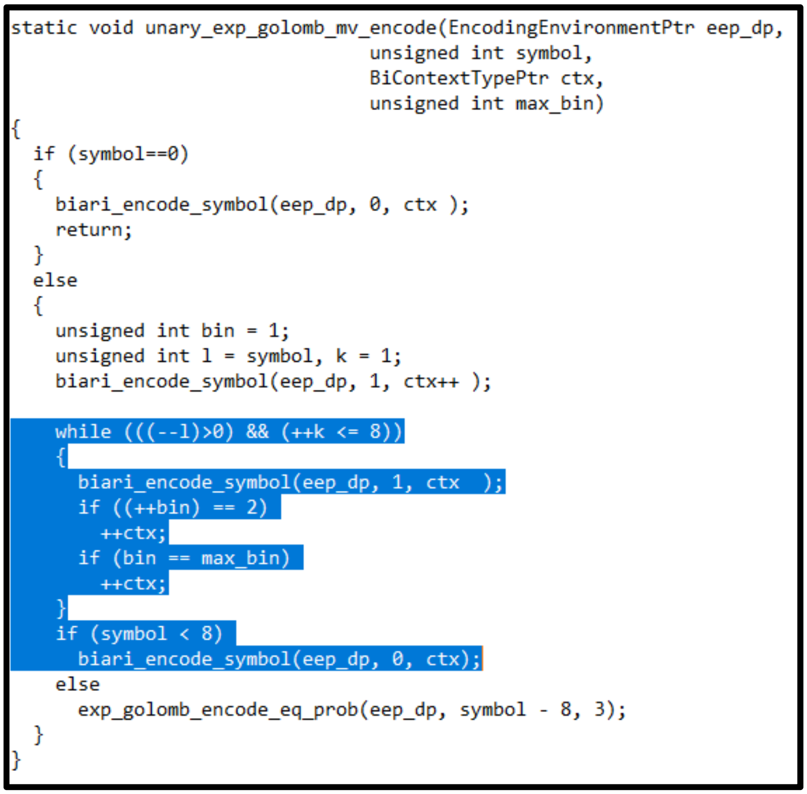
Source: unary_exp_golomb_mv_encode() function.



```
static void unary_exp_golomb_level_encode( EncodingEnvironmentPtr eep_dp,
                                           unsigned int symbol,
                                           BiContextTypePtr ctx)
{
  if (symbol==0)
  {
    biari_encode_symbol(eep_dp, 0, ctx );
    return;
  }
  else
  {
    unsigned int l=symbol;
    unsigned int k = 1;

    biari_encode_symbol(eep_dp, 1, ctx );
    while (((--l)>0) && (++k <= 13))
      biari_encode_symbol(eep_dp, 1, ctx);
    if (symbol < 13)
      biari_encode_symbol(eep_dp, 0, ctx);
    else
      exp_golomb_encode_eq_prob(eep_dp,symbol - 13, 0);
  }
}
```

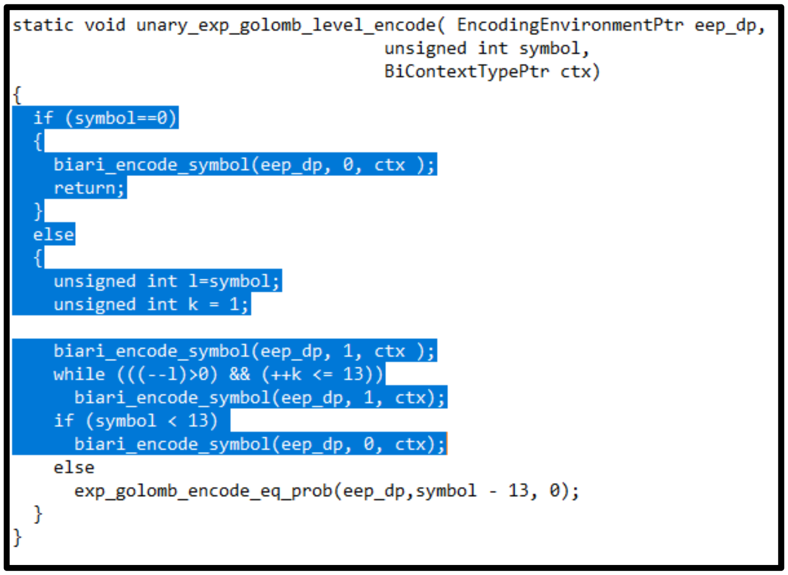
Source: unary_exp_golomb_level_encode() function.

227. On information and belief, the Netflix video encoding pipeline includes a “means for constructing a codeword using a unary binarization if said code symbol index value is less than said threshold value.” For example, in the H.264.2 reference software, the functions unary_exp_golomb_mv_encode() and unary_exp_golomb_level_encode() perform unary binarization of various syntax elements when the symbol index value is less than a threshold value (said threshold value is 8 and 13, respectively, in the excepts below).



```
static void unary_exp_golomb_mv_encode(EncodingEnvironmentPtr eep_dp,
                                unsigned int symbol,
                                BiContextTypePtr ctx,
                                unsigned int max_bin)
{
  if (symbol==0)
  {
    biari_encode_symbol(eep_dp, 0, ctx );
    return;
  }
  else
  {
    unsigned int bin = 1;
    unsigned int l = symbol, k = 1;
    biari_encode_symbol(eep_dp, 1, ctx++ );

    while (((--l)>0) && (++k <= 8))
    {
      biari_encode_symbol(eep_dp, 1, ctx  );
      if ((++bin) == 2)
        ++ctx;
      if (bin == max_bin)
        ++ctx;
    }
    if (symbol < 8)
      biari_encode_symbol(eep_dp, 0, ctx);
    else
      exp_golomb_encode_eq_prob(eep_dp, symbol - 8, 3);
  }
}
```

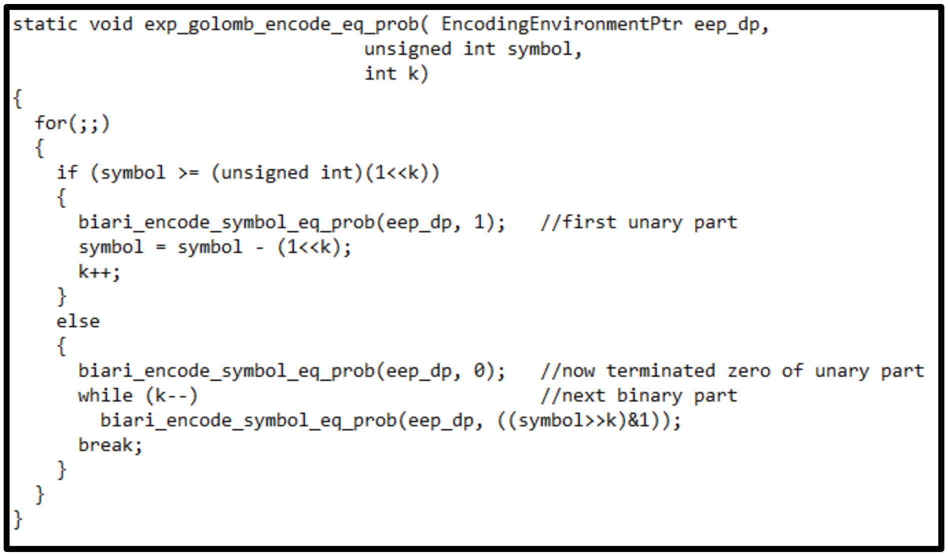
Source: unary_exp_golomb_mv_encode() function.



```
static void unary_exp_golomb_level_encode( EncodingEnvironmentPtr eep_dp,
                                    unsigned int symbol,
                                    BiContextTypePtr ctx)
{
  if (symbol==0)
  {
    biari_encode_symbol(eep_dp, 0, ctx );
    return;
  }
  else
  {
    unsigned int l=symbol;
    unsigned int k = 1;

    biari_encode_symbol(eep_dp, 1, ctx );
    while (((--l)>0) && (++k <= 13))
      biari_encode_symbol(eep_dp, 1, ctx);
    if (symbol < 13)
      biari_encode_symbol(eep_dp, 0, ctx);
    else
      exp_golomb_encode_eq_prob(eep_dp,symbol - 13, 0);
  }
}
```

Source: unary_exp_golomb_level_encode() function.

228. On information and belief, the Netflix video encoding pipeline also includes a "means for constructing a codeword using an exp-Golomb binarization if said code symbol index value is not less than said threshold value." For instance, functions unary_exp_golomb_mv_encode() and unary_exp_golomb_level_encode() in the H.264.2 reference software both call the function exp_golomb_encode_eq_prob() when the code symbol value is not less than the above-described threshold in order to construct the remainder of the codeword using exp-Golomb binarization.



```
static void exp_golomb_encode_eq_prob( EncodingEnvironmentPtr eep_dp,
                                       unsigned int symbol,
                                       int k)
{
  for(;;)
  {
    if (symbol >= (unsigned int)(1<<k))
    {
      biari_encode_symbol_eq_prob(eep_dp, 1);   //first unary part
      symbol = symbol - (1<<k);
      k++;
    }
    else
    {
      biari_encode_symbol_eq_prob(eep_dp, 0);   //now terminated zero of unary part
      while (k--)                               //next binary part
        biari_encode_symbol_eq_prob(eep_dp, ((symbol>>k)&1));
      break;
    }
  }
}
```

Source: exp_golomb_encode_eq_prob() function.

229. On information and belief, Netflix's video encoding pipeline also infringes claim 3 of the '387 Patent through its use of H.265 encoding. On information and belief, the reference software associated with that format proposes a binarization process that operates in substantially the same way as described above with regard to the H.264.2 reference software. On information and belief, Netflix uses the approach to binarization proposed by the H.265.2 reference software in encoding content files to the H.265 format.

230. At least as of on or around September 26, 2019, when the Broadcom Entities informed Netflix of its infringement of the ’387 Patent, and by no later than the date of the original complaint in this matter, Netflix has had knowledge of the ’387 Patent and the infringement thereof by its video encoding pipeline.

231. Netflix’s knowing and willful infringement of the ’387 Patent has caused and continues to cause damage to Broadcom Corp., and Broadcom Corp. is entitled to recover damages sustained as a result of Netflix’s wrongful acts in an amount subject to proof at trial.

EIGHTH CLAIM FOR RELIEF

(Infringement of U.S. Patent No. 6,982,663)

232. The Broadcom Entities reallege and incorporate by reference the allegations of paragraphs 1-231 set forth above.

233. The ’663 Patent, entitled “Method and System for Symbol Binarization,” was duly and legally issued on January 3, 2006 from a patent application filed on Jul. 10, 2002, naming Lowell Winger as the inventor. A copy of the ’663 Patent is attached hereto as **Exhibit H**.

234. The ’663 Patent is assigned to Broadcom Corp., which holds all substantial rights, title, and interest in and to the ’663 Patent.

235. Pursuant to 35 U.S.C. § 282, the ’663 Patent is presumed valid.

236. The ’663 Patent originates from the same specification as the ’387 Patent. Like the ’387 Patent, the ’663 Patent generally concerns an improvement in the way a computer system compresses visual and audio data. Specifically, the

patent is "directed to an improved method for the binarization of data in an MPEG data stream."[82]

237. As the '663 Patent explains, MPEG refers to a family of international standards developed by the Motion Picture Expert Group that specify how to represent visual and audio information in a compressed digital format.[83] The MPEG formats make it possible to "represent[] a video signal with data roughly 1/50th the size of the original uncompressed video, while still maintaining good visual quality."[84] The MPEG formats achieve such high compression by taking advantage of the fact that many images in a video stream do not change significantly from picture to picture, and if they do change, the differences from one picture to the next are often simple.[85] Storing and transmitting only the changes, instead of entire pictures, results in considerable savings in data transmission.[86]

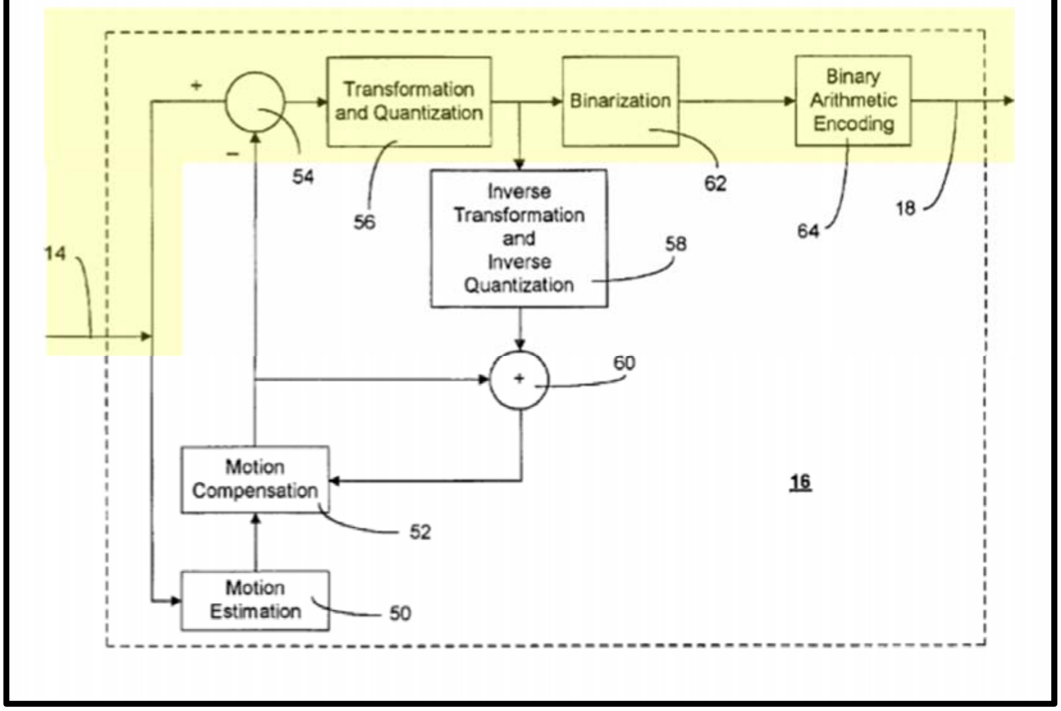
238. In practice, this compression technique is accomplished in a number of steps. First, pixel differences between the pictures are "transformed into frequency coefficients, and then quantized to further reduce the data transmission."[87] The '663 Patent refers to the resulting transformed and quantized coefficients as "symbols."[88] Second, the transformed-quantized symbols are "binarized" to create binary representations of each symbol in the form of a "codeword."[89] Third, the binarized codewords are entropy encoded "to reduce the number of bits per symbol

[82] '663 Patent, Abstract; *see also id.* at 1:7-11.
[83] *Id.* at 1:15-30.
[84] *Id.* at 1:38-41.
[85] *Id.* at 3:31-36.
[86] *Id.* at 1:51-54; 3:35-39.
[87] *Id.* at 1:55-58; 3:41-44.
[88] *See, e.g., id.* at 1:55-64, 4:1-4, 4:45-54, 5:10-46, Table 1.
[89] *Id.* at 4:1-4.

without introducing any additional video signal distortion."[90] The patent explains that several types of codecs[91] exist for performing the entropy encoding, "[o]ne of the most efficient of which is the family of binary arithmetic encoders (BACs)."[92] As the name implies, BACs operate only on binary valued data, which is why the symbols must be binarized before they can be entropy encoded.[93]

239. Figure 2 of the '663 Patent—the relevant portion of which is highlighted below—is a block diagram of an encoder that carries out this encoding process.

FIG. 2


+
–
Transformation and Quantization
Binarization
Binary Arithmetic Encoding
54
56
62
64
18
Inverse Transformation and Inverse Quantization
58
14
+
60
Motion Compensation
52
16
Motion Estimation
50

Source: '663 Patent, Fig. 2 (highlighting added).

[90] *Id.* at 1:59-63, 4:26-36.
[91] A codec is a device or computer program for encoding or decoding a digital stream or signal.
[92] '663 Patent, 4:34-36.
[93] *Id.* at 4:42-47.

240. Figure 2 depicts that a source video stream (14) enters the encoder (16) at the left and passes to a "combination" module (54), which assembles data related to pixel differences between pictures in the video stream.[94] The output of the combination module passes to the next module (56), where it is transformed and quantized. Symbols created by module (56) pass to the binarization module (62), which creates binary codewords that represent the symbols. The codewords next pass to binary arithmetic encoding module (64), where they are entropy encoded. And finally, the encoded bitstream (18) exits the encoder at the right.[95]

241. The '663 Patent is directed, in part, to the step in this process that occurs in the binarization module 62.[96] In the prior art, several techniques were available for binarizing the symbols, including, for example: unary, binary, Golomb, and exp-Golomb binarization.[97] The patent explains that those techniques each have certain strengths and weaknesses. Unary binarization, for example, generates codewords that are distinguishable from one another but that can be exceptionally long. Specifically, "[u]nary binarization consists of a number of binary 1s equal to an index for a symbol followed by a zero…."[98] Thus, a symbol index value of "1" results in a 2-bit codeword, namely "10." A symbol index value of "2" results in the 3-bit codeword "110," and so on. Thus, each codeword is easily distinguishable from the others, but the number of binary values can be quite large—encoding a large symbol index may require tens of thousands of bits.[99]

[94] *Id.* at 3:50-58.
[95] *Id.* at 3:57-4:8.
[96] *Id.* at 4:1-2.
[97] *Id.* at 4:46-47.
[98] *Id.* at 4:48-49.
[99] *Id.* at 5:22-5:45.

242. Exp-Golomb binarization, on the other hand, greatly reduces the maximum possible size of the codewords, but "it does not permit codewords with a small symbol index (other than index 0) to be uniquely distinguished from codewords with larger symbol indices."[100]

243. Thus, there was a need for a method and system that could exploit the most valuable properties of the unary and exp-Golomb binarizations.[101] The invention described in the '663 Patent meets that need:

> The present invention provides a binarization that retains the most valuable properties of the unary and exp-Golomb binarizations. That is, small codewords are distinguishable as with a unary binarization, while large codewords have their binarization limited to a reasonable length. By doing so, the present invention provides a binarization that reduces the complexity and the bitrate/size for compressing and decompressing video, images, and signals that are compressed using binary arithmetic encoding for entropy encoding.[102]

244. The '663 Patent claims methods and systems utilizing a combination of unary and exp-Golomb binarization for encoding and decoding digital video data. Independent claim 12, for example, recites:

> A method for generating a codeword from an index value for digital video encoding, comprising the steps of:
>
> (A) generating a first pattern in a first portion of said codeword in response to said index value being at least as great as a threshold;
>
> (B) generating a second pattern in a second portion of said codeword following said first portion representing an offset of said index value above said threshold; and

[100] *Id.* at 6:14-17.
[101] *Id.* at 2:1-12.
[102] *Id.* at 6:19-28.

> (C) generating a third pattern in a third portion of said codeword following said second portion representing a value of said index value above said offset.

245. As in the ’387 Patent, the methods and systems described in the ’663 Patent improve the functionality of computer systems by improving the way they compress and process video and audio data.

246. Notably, the Hon. James V. Selna of this District previously held that the claims of the ’663 Patent are patent-eligible under 35 U.S.C. § 101. In doing so, Judge Selna concluded that claim 12, and others in the ’663 Patent, are “directed to improving digital video decoding” and, thus, “are not directed to abstract ideas.”[103]

247. Upon information and belief, Netflix directly infringes at least claim 12 of the ’663 Patent at least in the exemplary manner described below.

248. Figure 5 of the ’663 Patent, which depicts Table 3, demonstrates a particular instance of the hybrid unary-exp-Golomb codes described in that patent. As the ’663 Patent explains, “Table 3 illustrates a binarization that is particularly appropriate for the binarization of quarter pixel motion vector residual magnitudes of MPEG-AVC/H.264.”[104] The patent describes how, upon reaching the threshold at which unary to exp-Golomb switching occurs (N=64 for Table 3), the index comprises three parts: (1) the initial prefix (highlighted in blue below); (2) a unary representation appended to the initial prefix to form the unary prefix (highlighted in red below); and (3) the exp-Golumb suffix (highlighted in green below).[105]

[103] *Broadcom Corp., et al. v. Sony Corp., et al.*, SAVC 16-1052 JVS (JCGx), p. 9 (C.D.C.A. Oct. 5, 2016).

[104] ’663 Patent, 6:35-40.

[105] *See id.* at 6:44-63.

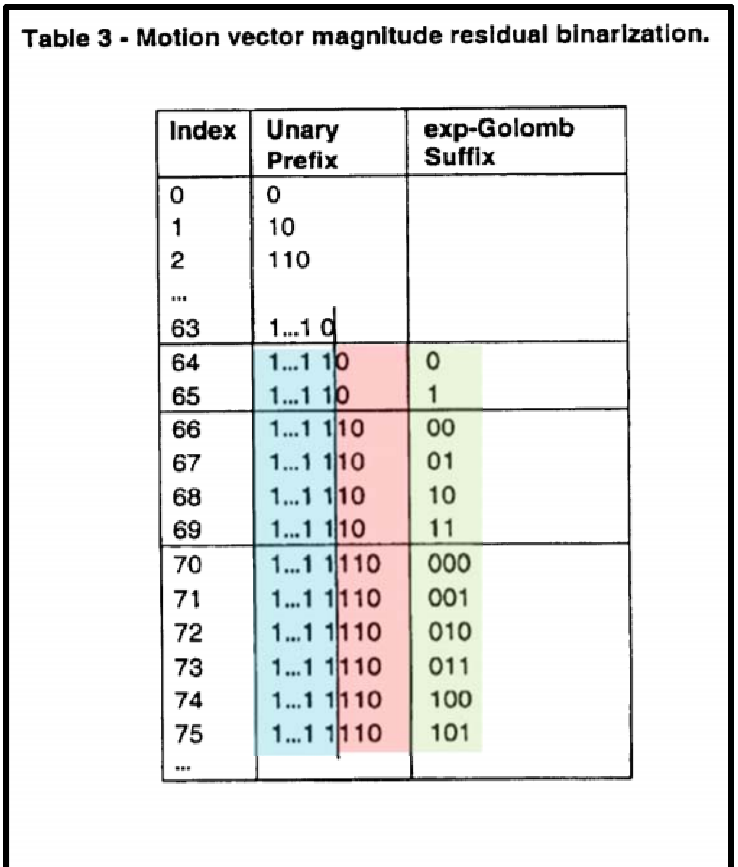


Table 3 - Motion vector magnitude residual binarization.

Index	Unary Prefix	exp-Golomb Suffix
0	0	
1	10	
2	110	
…		
63	1…1 0	
64	1…1 10	0
65	1…1 10	1
66	1…1 110	00
67	1…1 110	01
68	1…1 110	10
69	1…1 110	11
70	1…1 1110	000
71	1…1 1110	001
72	1…1 1110	010
73	1…1 1110	011
74	1…1 1110	100
75	1…1 1110	101
…		

FIG. 5

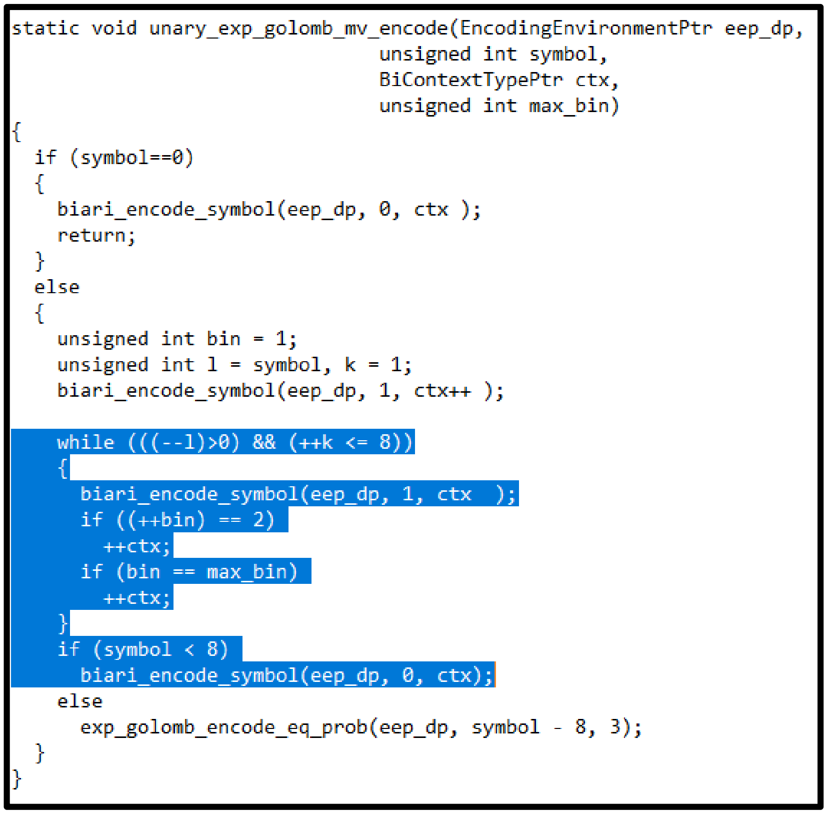
Source: ’663 Patent, Fig. 5.

249. As explained above, Netflix developed, operates, and uses a video encoding pipeline to encode its film and TV content in a variety of digital formats, including H.264 and H.265.

250. The Netflix video encoding system practices a “method for generating a codeword from an index value for digital video encoding.” For instance, with regard to the H.264 format used by Netflix, the H.264 documentation explains how the format employs a “concatenated unary/k-th order Exp-Golomb (UEGk) binarization process.” This process generates codewords from index values. On information and belief, Netflix uses the method for UEG(k) encoding set forth in

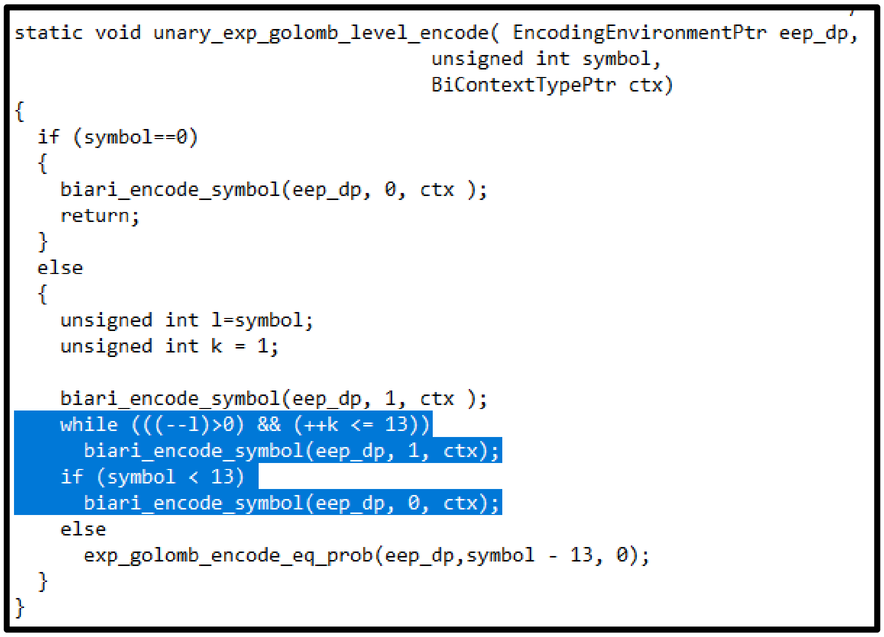
the H.264.2 reference software, which serves as an aid for the study and implementation of H.264 video coding.

251. On information and belief, the Netflix video encoding pipeline practices the step of "generating a first pattern in a first portion of said codeword in response to said index value being at least as great as a threshold." For instance, the H.264.2 reference software features the functions unary_exp_golomb_mv_encode() and unary_exp_golomb_level_encode(), which are responsible—in part—for generating codewords from index values during the encoding process. Where the index value meets or exceeds a certain threshold (8 for unary_exp_golomb_mv_encode() and 13 for unary_exp_golomb_level_encode()), these functions populate the first portion of the codeword with a pattern representing an initial prefix.



```
static void unary_exp_golomb_mv_encode(EncodingEnvironmentPtr eep_dp,
                                unsigned int symbol,
                                BiContextTypePtr ctx,
                                unsigned int max_bin)
{
  if (symbol==0)
  {
    biari_encode_symbol(eep_dp, 0, ctx );
    return;
  }
  else
  {
    unsigned int bin = 1;
    unsigned int l = symbol, k = 1;
    biari_encode_symbol(eep_dp, 1, ctx++ );

    while (((--l)>0) && (++k <= 8))
    {
      biari_encode_symbol(eep_dp, 1, ctx  );
      if ((++bin) == 2)
        ++ctx;
      if (bin == max_bin)
        ++ctx;
    }
    if (symbol < 8)
      biari_encode_symbol(eep_dp, 0, ctx);
    else
      exp_golomb_encode_eq_prob(eep_dp, symbol - 8, 3);
  }
}
```

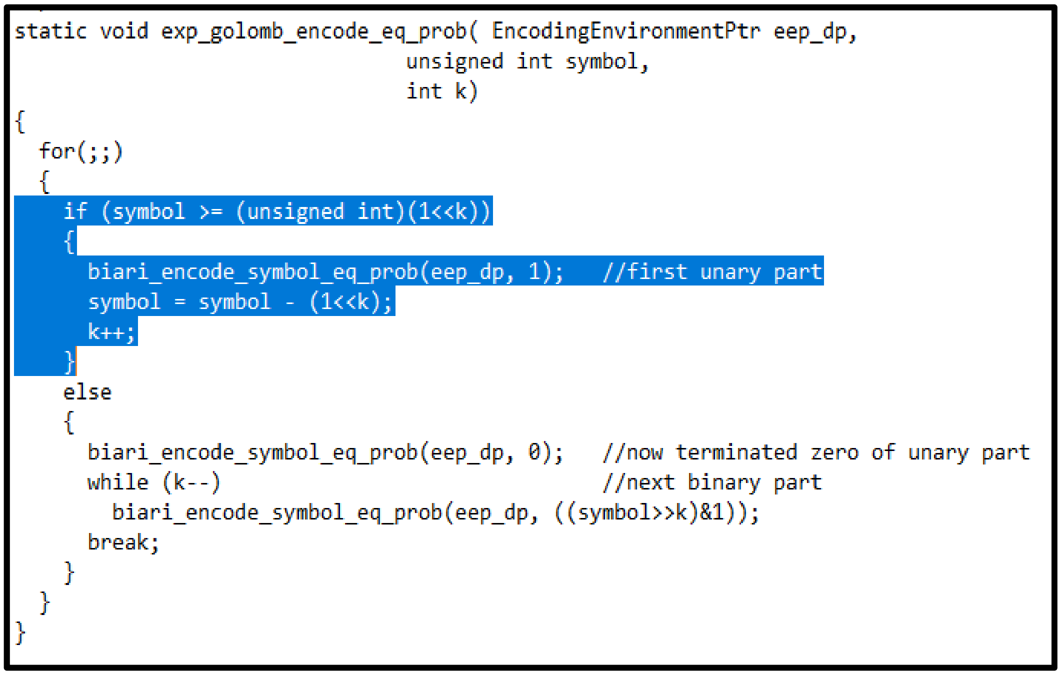
Source: unary_exp_golomb_mv_encode() function.



```
static void unary_exp_golomb_level_encode( EncodingEnvironmentPtr eep_dp,
                                           unsigned int symbol,
                                           BiContextTypePtr ctx)
{
  if (symbol==0)
  {
    biari_encode_symbol(eep_dp, 0, ctx );
    return;
  }
  else
  {
    unsigned int l=symbol;
    unsigned int k = 1;

    biari_encode_symbol(eep_dp, 1, ctx );
    while (((--l)>0) && (++k <= 13))
      biari_encode_symbol(eep_dp, 1, ctx);
    if (symbol < 13)
      biari_encode_symbol(eep_dp, 0, ctx);
    else
      exp_golomb_encode_eq_prob(eep_dp,symbol - 13, 0);
  }
}
```

Source: unary_exp_golomb_level_encode() function.

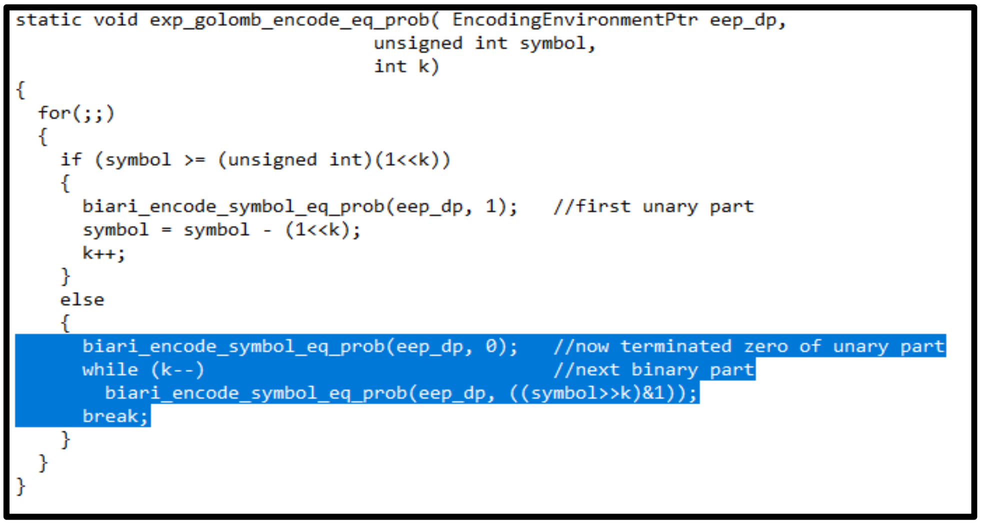
252. On information and belief, the Netflix video encoding system also "generat[es] a second pattern in a second portion of said codeword following said first portion representing an offset of said index value above said threshold." For example, both the unary_exp_golomb_mv_encode() and unary_exp_golomb_level_encode() functions call the function exp_golomb_encode_eq_prob() when the index value exceeds the thresholds described above. The exp_golomb_encode_eq_prob() function generates a second portion of the codeword—the unary representation appended to the initial prefix to form the unary prefix—when the index value exceeds the thresholds.



```
static void exp_golomb_encode_eq_prob( EncodingEnvironmentPtr eep_dp,
                                unsigned int symbol,
                                int k)
{
  for(;;)
  {
    if (symbol >= (unsigned int)(1<<k))
    {
      biari_encode_symbol_eq_prob(eep_dp, 1);   //first unary part
      symbol = symbol - (1<<k);
      k++;
    }
    else
    {
      biari_encode_symbol_eq_prob(eep_dp, 0);   //now terminated zero of unary part
      while (k--)                               //next binary part
        biari_encode_symbol_eq_prob(eep_dp, ((symbol>>k)&1));
      break;
    }
  }
}
```

Source: exp_golomb_encode_eq_prob() function.

253. Further, on information and belief, the Netflix video encoding system "generat[es] a third pattern in a third portion of said codeword following said second portion representing a value of said index value above said offset." For example, after the unary prefix described above, the exp_golomb_encode_eq_prob() function generates a third portion of the codeword—the exp-Golumb suffix—which captures the value of the index above the offset described above.



```
static void exp_golomb_encode_eq_prob( EncodingEnvironmentPtr eep_dp,
                                        unsigned int symbol,
                                        int k)
{
  for(;;)
  {
    if (symbol >= (unsigned int)(1<<k))
    {
      biari_encode_symbol_eq_prob(eep_dp, 1);   //first unary part
      symbol = symbol - (1<<k);
      k++;
    }
    else
    {
      biari_encode_symbol_eq_prob(eep_dp, 0);   //now terminated zero of unary part
      while (k--)                               //next binary part
        biari_encode_symbol_eq_prob(eep_dp, ((symbol>>k)&1));
      break;
    }
  }
}
```

Source: exp_golomb_encode_eq_prob() function.

254. On information and belief, Netflix's video encoding pipeline also infringes claim 12 of the '663 Patent through its use of H.265 encoding. On information and belief, the reference software associated with that format proposes a binarization process that operates in substantially the same way as described above with regard to the H.264.2 reference software. On information and belief, Netflix uses the approach to binarization proposed by the H.265.2 reference software in encoding content files to the H.265 format.

255. At least as of on or around September 26, 2019, when the Broadcom Entities informed Netflix of its infringement of the '663 Patent, and by no later than the filing of the original complaint in this matter, Netflix has had knowledge of the '663 Patent and the infringement thereof by its encoding pipeline.

256. Netflix's knowing and willful infringement of the '663 Patent has caused and continues to cause damage to Broadcom Corp., and Broadcom Corp. is entitled to recover damages sustained as a result of Netflix's wrongful acts in an amount subject to proof at trial.

NINTH CLAIM FOR RELIEF

(Infringement of U.S. Patent No. 9,332,283)

257. The Broadcom Entities reallege and incorporate by reference the allegations of paragraphs 1-256 set forth above.

258. The ’283 Patent, entitled “Signaling of Prediction Size Unit in Accordance with Video Coding,” was duly and legally issued on May 3, 2016 from a patent application filed on June 14, 2012, with Peisong Chen, Brian Heng, and Wade Wan as the named inventors. A copy of the ’283 Patent is attached hereto as **Exhibit I**.

259. The ’283 Patent claims priority from U.S. Provisional Application No. 60/539,948, filed on September 27, 2011.

260. The ’283 Patent is assigned to Broadcom Corp., which holds all substantial rights, title, and interest in and to the ’283 Patent.

261. Pursuant to 35 U.S.C. § 282, the ’283 Patent is presumed valid.

262. The ’283 Patent generally concerns an improved system for encoding and decoding video content that ensures a high quality output when transmitting that content to viewers. Specifically, the patent relates to an improved method for encoding video content using a process known as binarization in order to transmit that information to users more efficiently.

263. At the time of the inventions claimed in the ’283 Patent, persons engaged in developing next-generation video encoding technologies were looking to take advantage of ongoing innovations in parallel processing power and increased video resolutions, which could make performing complex encoding operations more efficient and lead to improved user experiences. Video encoding is

a multi-step process that begins with an input video signal, and results in an output bitstream of encoded video data. As discussed above, between the input of the video signal and the output of the encoded video data the encoding process includes various operations that are performed on constituent portions of the input video signal to create a video stream according to a particular encoding format.

264. As the patent explains, among the advances at the time was the use of "predictive" (P) slices and "bi-predictive" (B) slices as components of the video data being processed.[106] P slices and B slices, in turn, are comprised of smaller components, including "coding tree units" and "coding units." As described in the '283 Patent, coding units can be "encoded" for different types of "prediction" processing, namely, "inter-prediction" or "intra-prediction."[107] Subsequently, "prediction units" (PU) can be encoded for different "partition modes," to be used in the intra- or inter-prediction processing.

265. Generally speaking, prediction and partition modes are encoded using binary "codewords." These codewords can be generated using a "binary tree." In general terms, a binary tree is a data structure that can be used to represent data and associate it with a corresponding bit sequence and vice versa. In this context, a "binary tree" data structure is used to create a sequence of binary numbers based on the selection of a "1" or a "0" at different positions (sometimes referred to as "nodes") in the binary tree, starting at the beginning ("root"). Traversing the binary tree from the root to a "leaf" (an endpoint) results in a bit sequence that corresponds

[106] '283 Patent, at 17:24-48.

[107] *See, e.g.*, *id.* at 17:9-12.

to a specific encoding.[108] Figure 13 of the ’283 Patent, for example, represents binary trees as follows:

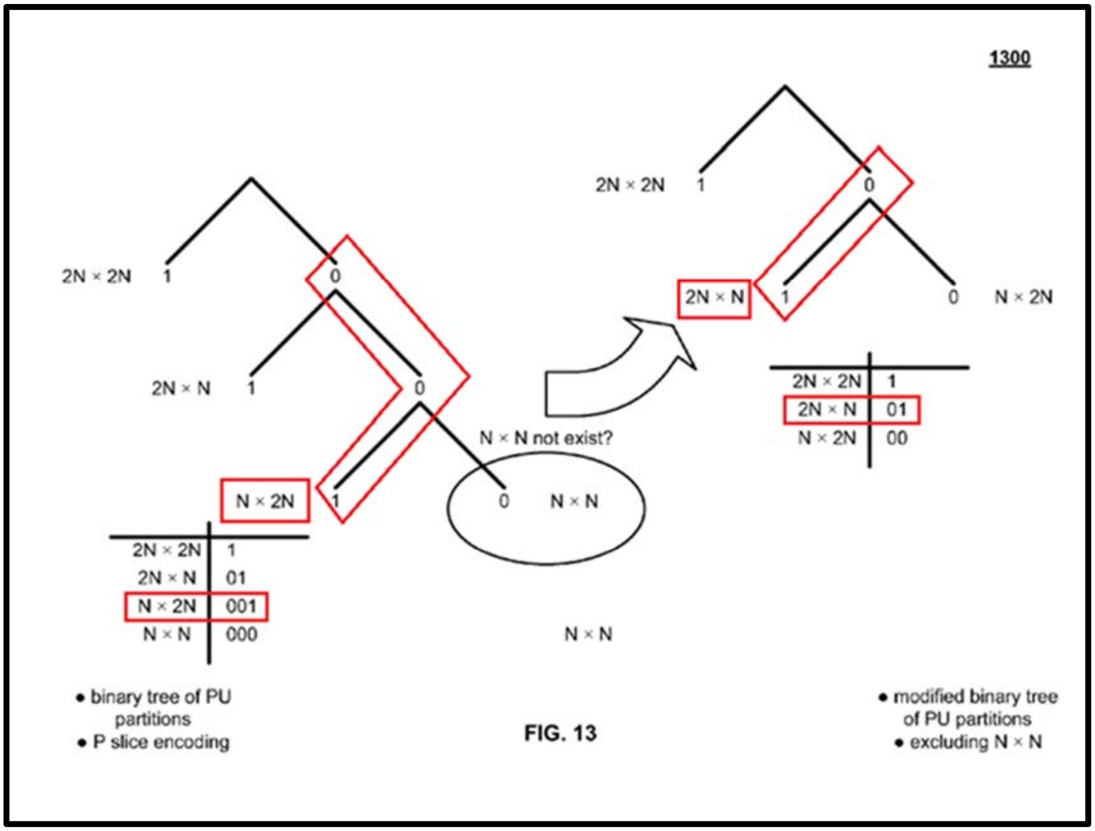


FIG. 13

Source: ’283 Patent, Fig. 13 (annotations added).

In the left side of the exemplary image above, the annotated binary tree generates a codeword of “001,” which is associated with a partition mode of “Nx2N.”[109]

266. The use of P slices and B slices in an encoding protocol can introduce inefficiencies when encoding coding units of the different slice types. This is because at the time of the invention of the ’283 Patent, the encoding of P and B slices required separate codewords generated using different binary trees, which resulted in extra burdens on the encoding system and higher overhead.[110]

[108] *See id.* at Figs. 13, 14, 15.
[109] *Id.* at Fig. 13 (annotations added).
[110] *Id.* at 19:21-35; 20:19-30.

267. Thus, there was a need for a method and system that could reduce this inefficiency.[111] The invention described in the ’283 Patent meets that need. Specifically, the ’283 Patent’s system and method for using “only a singular codebook . . . for both processing of the B slices and P slices,” may be employed to provide a “very efficient implementation.”[112] The methods and systems described in the ’283 Patent thus improve the functionality of computer systems by improving the way they compress and process video and audio data.

268. The inventions described and claimed in the ’283 Patent include encoding using two syntax elements derived from a single binary tree that can be applied to both P slices and B slices to indicate (1) whether inter-prediction or intra-prediction applies to a selected coding unit, and (2) the prediction unit partition mode that applies based on whether or not the selected coding unit is the smallest coding unit (SCU) having a prediction unit size NxN.

269. The ’283 Patent claims methods and systems that use a single binary tree to encode coding unit (CU) prediction when processing P slices and B slices for digital video data. Independent claim 1, for example, recites:

> A video processing device comprising:
>
> a video encoder configured to:
>
> encode an input video signal to generate an output bitstream;
>
> employ a single binary tree when processing at least one P slice and at least one B slice to generate the output bitstream, wherein the at least one P slice is used for unidirectional prediction forward or behind in at least one frame sequence, and wherein the at least one B slice is

[111] ’283 Patent at 18:55-19:12.
[112] *Id*. at 18:60-63.

> used for bidirectional prediction both forward and behind in the at least one frame sequence;
>
> employ the single binary tree to encode coding unit (CU) prediction based on a selected CU that is selected from a plurality of CUs when generating a first syntax element for both the at least one P slice and the at least one B slice that undergo entropy encoding to generate the output bitstream, wherein the first syntax element specifies intra-prediction processing or inter-prediction processing for the selected CU; and
>
> employ the single binary tree to encode prediction unit (PU) partition mode based on the selected CU when generating a second syntax element for both the at least one P slice and the at least one B slice that undergo the entropy encoding to generate to generate [sic] the output bitstream, wherein the second syntax element specifies the PU partition mode for the selected CU, wherein the PU partition mode is based on a size N×N PU when the selected CU is a smallest CU (SCU) of the plurality of CUs and is based on a different size PU than the size N×N PU when the selected CU is another CU than the SCU of the plurality of CUs, wherein N is a positive integer.

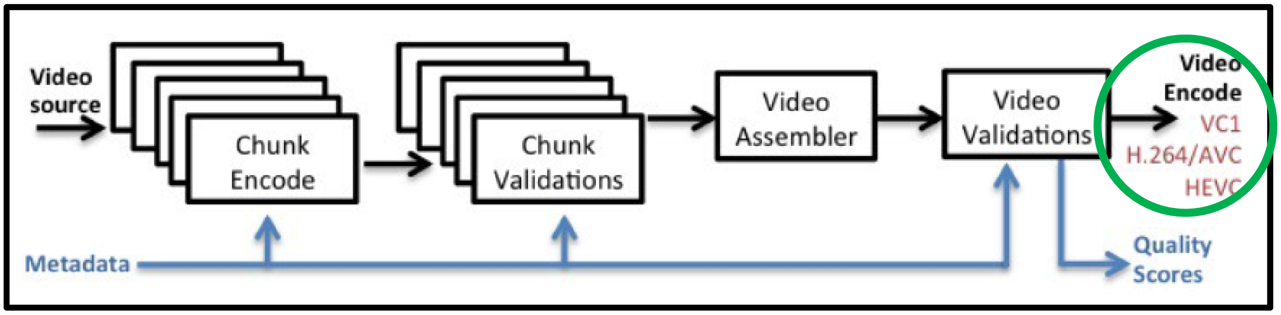
270. Netflix directly infringes at least claim 1 of the ’283 Patent at least in the exemplary manner described below.

271. Netflix developed, operates, and uses a “video encoding pipeline”, *i.e.*, a series of video processing applications. Netflix uses its video encoding pipeline to generate encoded video files in a variety of formats, which it then uses to stream movie and TV content to its subscribers. As Netflix explains:

> We ingest high quality video sources and generate video encodes of various codec profiles, at multiple quality representations per profile. The encodes are packaged and then deployed to a content delivery network for streaming. During a streaming session, the client requests the encodes it can play and adaptively switches among

quality levels based on network conditions.[113]

272. Among other encoding formats, Netflix uses its video encoding pipeline to generate content in the H.265 format, also known as High Efficiency Video Coding or "HEVC" ("H.265").



Source: https://medium.com/netflix-techblog/high-quality-video-encoding-at-scale-d159db052746.

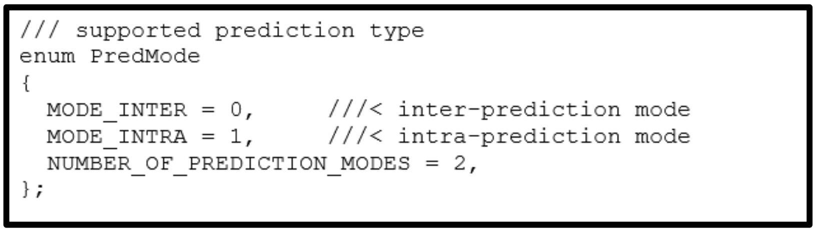
273. On information and belief, the Netflix video encoder includes a binarization system that employs a "single binary tree." For example, the H.265 format employs a "binary tree" in the processing (i.e., encoding) of P slices and B slices. Specifically, the H.265 format utilizes coding schemes that define a unique mapping of syntax element values to sequences of binary symbols, which are interpreted in terms of a binary code tree.[114]

274. On information and belief, the Netflix video encoder employs the binary tree to generate a "first syntax element" of a coding unit, wherein the "first syntax element" specifies intra-prediction processing or inter-prediction processing for the selected coding unit. For example, H.265 reference software features the

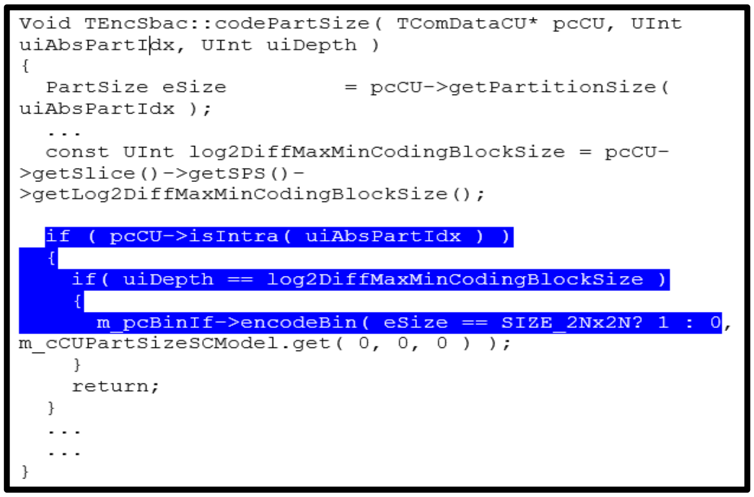
[113] https://medium.com/netflix-techblog/high-quality-video-encoding-at-scale-d159db052746.

[114] *See, e.g.*, *ITU-T Recommendation H.265: High Efficiency Video Coding*, at § 7.3.8.5 (November 2019) (available at https://www.itu.int/rec/T-REC-H.265). ("H.265 Recommendation") (describing binary tree coding syntax using pseudo code).

parameter PredMode which specifies the "prediction mode" (i.e., intra-prediction processing or inter-prediction processing) of a selected coding unit. The function pcCU->isIntra() then returns either 1 or 0 depending on whether intra- or inter-prediction mode is selected for the particular coding unit. If the selected coding unit pcCU was encoded using intra-prediction mode (i.e., pcCU->isIntra() is TRUE) the encodeBin() function codes 1. If inter-prediction mode is selected, the encodeBin() codes 0.[115]



```
/// supported prediction type
enum PredMode
{
  MODE_INTER = 0,      ///< inter-prediction mode
  MODE_INTRA = 1,      ///< intra-prediction mode
  NUMBER_OF_PREDICTION_MODES = 2,
};
```



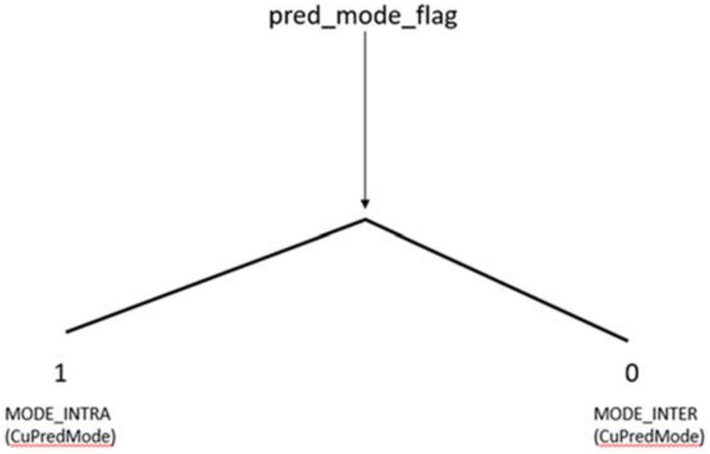
```
Void TEncSbac::codePartSize( TComDataCU* pcCU, UInt
uiAbsPartIdx, UInt uiDepth )
{
  PartSize eSize         = pcCU->getPartitionSize(
uiAbsPartIdx );
  ...
  const UInt log2DiffMaxMinCodingBlockSize = pcCU-
>getSlice()->getSPS()-
>getLog2DiffMaxMinCodingBlockSize();

  if ( pcCU->isIntra( uiAbsPartIdx ) )
  {
    if( uiDepth == log2DiffMaxMinCodingBlockSize )
    {
      m_pcBinIf->encodeBin( eSize == SIZE_2Nx2N? 1 : 0,
m_cCUPartSizeSCModel.get( 0, 0, 0 ) );
    }
    return;
  }
  ...
  ...
}
```

Source: H.265.2: Reference software for ITU-T H.265 High Efficiency Video Coding (December 2016) (available at https://www.itu.int/rec/T-REC-H.265.2).

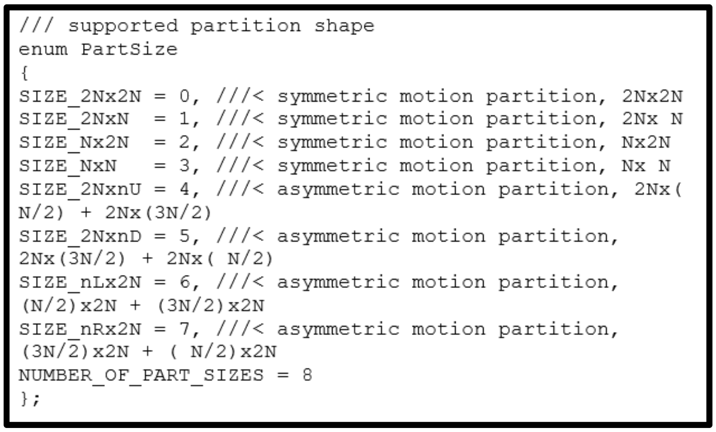
275. The binary tree that the Netflix video encoder employs to encode a coding unit prediction mode using a first syntax element for coding units for both P

[115] *See also*, H.265 Recommendation at § 7.4.9.5.

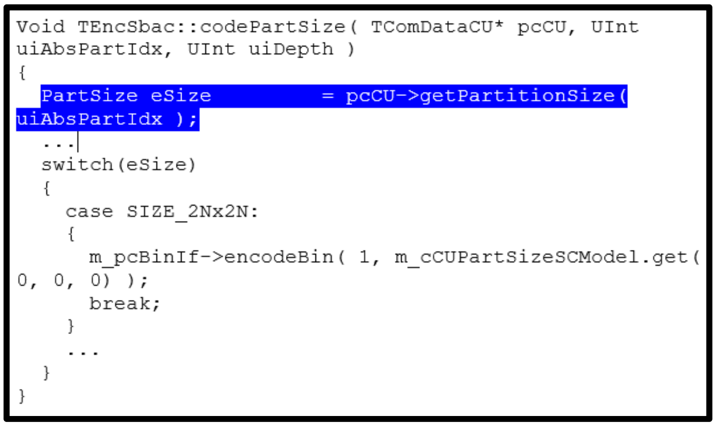
slices and B slices thus may be represented as follows, in which “1” codes for intra-prediction, and “0” codes for inter-prediction:



276. The Netflix video encoder is configured to employ the single binary tree to encode prediction unit (“PU”) partition mode based on the selected coding unit “when generating a second syntax element for both the at least one P slice and the at least one B slice,” wherein the second syntax element specifies the PU partition mode for the selected CU. For example, the H.265 reference software specifies the parameter “PartSize” and the function “getPartitionSize” which identifies the partition mode for the selected CU (“pcCU”). This particular example shows how in the case of a 2N×2N partition, the function encodeBin() generates the binary code “1”.



```
/// supported partition shape
enum PartSize
{
SIZE_2Nx2N = 0, ///< symmetric motion partition, 2Nx2N
SIZE_2NxN  = 1, ///< symmetric motion partition, 2Nx N
SIZE_Nx2N  = 2, ///< symmetric motion partition, Nx2N
SIZE_NxN   = 3, ///< symmetric motion partition, Nx N
SIZE_2NxnU = 4, ///< asymmetric motion partition, 2Nx(
N/2) + 2Nx(3N/2)
SIZE_2NxnD = 5, ///< asymmetric motion partition,
2Nx(3N/2) + 2Nx( N/2)
SIZE_nLx2N = 6, ///< asymmetric motion partition,
(N/2)x2N + (3N/2)x2N
SIZE_nRx2N = 7, ///< asymmetric motion partition,
(3N/2)x2N + ( N/2)x2N
NUMBER_OF_PART_SIZES = 8
};
```



```
Void TEncSbac::codePartSize( TComDataCU* pcCU, UInt
uiAbsPartIdx, UInt uiDepth )
{
  PartSize eSize         = pcCU->getPartitionSize(
uiAbsPartIdx );
  ...
  switch(eSize)
  {
    case SIZE_2Nx2N:
    {
      m_pcBinIf->encodeBin( 1, m_cCUPartSizeSCModel.get(
0, 0, 0) );
      break;
    }
    ...
  }
}
```

Source: H.265.2: Reference software for ITU-T H.265 High Efficiency Video Coding (December 2016) (available at https://www.itu.int/rec/T-REC-H.265.2).

277. As indicated in the Table 9-45 of the H.265 Recommendation, the partition mode for a particular coding unit is dependent on the prediction mode (CuPredMode) of the coding unit.[116] Table 9-45 depicts the first syntax element (CuPredMode) and the second syntax element (encoding for the partition mode) for different size coding unit cases. The illustration below, in conjunction with Table 9-45, depicts the single binary tree encoding both the prediction mode and the partition mode.

[116] *See* H.265 Recommendation, at Table 9-45.

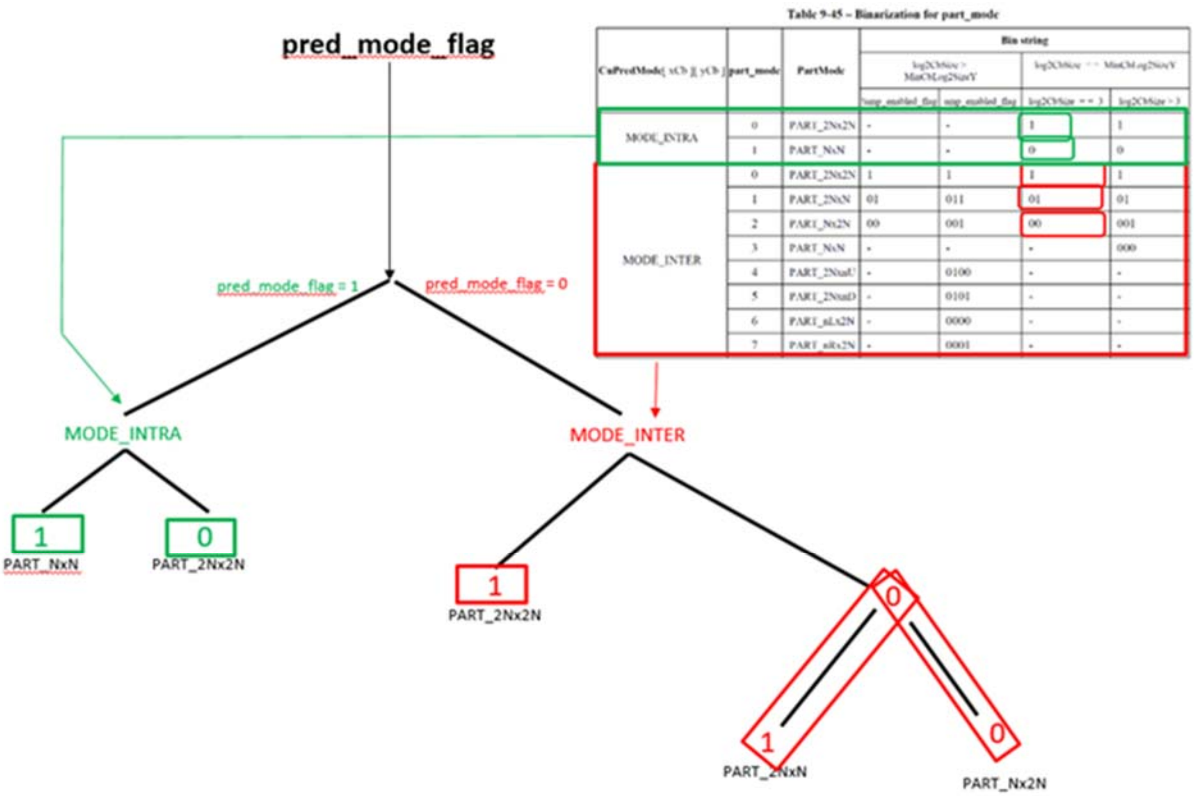


Table 9-45 – Binarization for part_mode

CuPredMode[xCb][yCb]	part_mode	PartMode	Bin string			
			log2CbSize > MinCbLog2SizeY		log2CbSize = = MinCbLog2SizeY	
			!amp_enabled_flag	amp_enabled_flag	log2CbSize = = 3	log2CbSize > 3
MODE_INTRA	0	PART_2Nx2N	-	-	1	1
	1	PART_NxN	-	-	0	0
MODE_INTER	0	PART_2Nx2N	1	1	1	1
	1	PART_2NxN	01	011	01	01
	2	PART_Nx2N	00	001	00	001
	3	PART_NxN	-	-	-	000
	4	PART_2NxnU	-	0100	-	-
	5	PART_2NxnD	-	0101	-	-
	6	PART_nLx2N	-	0000	-	-
	7	PART_nRx2N	-	0001	-	-

Source: H.265 Recommendation at Table 9-45 (annotations added).

278. Thus, on information and belief, the Netflix video encoder employs the single binary tree to encode prediction unit partition mode when generating a second syntax element, wherein the second syntax element specifies the partition mode for the selected coding unit. In the table above, the coding unit example is of a size "log2CbSize" equals "3".

279. As indicated in the image above, when the prediction mode is intra-prediction processing, the single binary tree encodes a second syntax element of either 1 or 0, which indicates a partition mode (PartMode (column 3)) of PART_2Nx2N or PART_NxN, respectively. Similarly, when the prediction mode is inter-prediction processing, the single binary tree for the example coding unit size

specifies a second syntax element of 1, 01, or 00, which indicates a partition mode of PART_2Nx2N, PART_2NxN, or PART_Nx2N, respectively.

280. On information and belief, the Neflix video encoder employs a PU partition mode that is "based on a size N×N PU when the selected CU is a smallest CU (SCU)" and "is based on a different size PU than the size N×N PU when the selected CU is another CU than the SCU." The H.265 format supports variable prediction block sizes from 64×64 down to 4×4 samples. The minimum size CU is assigned a size N×N Prediction Unit (PU) as a special case.[117] Thus, the prediction unit partition mode is based on a coding unit size of N×N when the coding unit is the smallest coding unit, and the prediction unit partition mode is based on a different size prediction unit when the coding unit is a size other than the smallest coding unit.[118] Thus, the Netflix video encoding system, as described above, practices at least claim 1 of the '283 Patent.

281. Indeed, Netflix has published studies in which it has quantified the benefits of H.265 encoding. Through its own testing, Netflix determined that H.265 encoders can achieve equivalent subjective reproduction quality as encoders that conform to H.264/MPEG-4 AVC while using approximately 50% less bit rate (i.e., the amount of data "bits" transmitted per second).[119] Thus, Netflix has specifically recognized benefits achieved by the inventions of the '283 Patent.

[117] *See e.g.*, *Overview of the High Efficiency Video Coding (HEVC) Standard*, IEEE Transactions On Circuits And Systems For Video Technology, Vol. 22, No. 12, (December 2012).

[118] *See* Table 9-45 at column "PartMode."

[119] https://netflixtechblog.com/a-large-scale-comparison-of-x264-x265-and-libvpx-a-sneak-peek-2e81e88f8b0f.

282. By no later than the date of the filing and service of the original complaint in this matter, Netflix has had knowledge of the ’283 Patent and the infringement thereof by its encoding system. Netflix’s infringement of the ’283 Patent, which is knowing and willful at least as of the filing and service of the original complaint in this matter, has caused and continues to cause damage to Broadcom Corp. Broadcom Corp. is entitled to recover damages sustained as a result of Netflix’s wrongful acts in an amount subject to proof at trial.

TENTH CLAIM FOR RELIEF

(Infringement of U.S. Patent No. 8,548,976)

283. The Broadcom Entities reallege and incorporate by reference the allegations of paragraphs 1-282 set forth above.

284. The ’976 Patent, entitled “Balancing Load Requests and Failovers Using a UDDI Proxy,” was duly and legally issued on October 1, 2013 from a patent application filed on May 19, 2005, with Christopher Betts and Tony Rogers as the named inventors. A copy of the ’976 Patent is attached hereto as **Exhibit J**

285. The ’976 Patent claims priority from U.S. Provisional Application No. 60/573,450, filed on May 21, 2004.

286. The ’976 Patent was assigned to Avago, which currently holds all substantial rights, title, and interest in and to the ’976 Patent.

287. Pursuant to 35 U.S.C. § 282, the ’976 Patent is presumed valid.

288. The ’976 Patent is directed to an improvement in the functionality of complex computer networks and how software services are delivered using the computational resources within those networks.

289. The ’976 Patent addresses specific technical challenges that arose in the computer networking environment as web services became an increasingly popular means of enabling access to software systems.

290. The specification of the ’976 Patent explains that “[w]eb services are software systems that can be identified by Universal Resource Identifiers (URI) in a fashion that is analogous to the way websites may be identified by Universal Resource Locators (URLs).”[120] Web services “enhance the way computers communicate with users and each other”[121] and, at the time, they were “quickly transforming the way modern businesses interact[ed] and share[d] information.”[122]

291. However, the use of web services presented a number of challenges, as the ’976 Patent describes. For instance, “before a software system can utilize the functionality of a web service, the software system should first be able to locate and connect to the web service,” a process known as “discovery and integration.”[123]

292. As another example, web services may be available to a large number of potential users, with some being accessible worldwide via the internet to a wide variety of users and software systems operating on a wide variety of platforms.[124] “Because of the very large number of potential web service users and the complexity of many web services, web services can push even the most capable servers running the web services to their limits.”[125] Over-loaded servers may not

[120] ’976 Patent, 1:19-23.
[121] *Id.* at 1: 26-27.
[122] *Id.* at 1:18-19.
[123] *Id.* at 1:34-38.
[124] *Id.* at 1:47-59.
[125] *Id.* at 1:60-63.

function properly, may stop handling web service requests, and may cease functioning altogether.[126]

293. As the ’976 Patent specification explains, one option for addressing the issue of excess load on a server from a web service is to spread the load across “multiple servers all working toward processing web service requests.”[127] However, this requires a system for balancing the load among the multiple servers.[128]

294. The ’976 Patent further explains that the web service provider may also wish to employ failover servers—“redundant or standby server[s] in the event the primary server fails.”[129] These failovers “may be used to ensure the continued offering of web services in any number of circumstances that may render the primary server non-functional.”[130]

295. The ’976 Patent claims specific, novel ways to address these technical challenges through methods, systems, and apparatuses that enable the efficient, effective provision of web services using multiple servers.

296. The methods, systems, and apparatuses described in the ’976 Patent improve the functionality of a networked system of computing resources by enabling the efficient, effective use of those resources to provide software services.

297. Claim 14 of the ’976 Patent reads as follows:

> A computer system comprising:
>
> a processor; and a computer recording medium including

[126] *Id.* at 1:66-2:3.
[127] *Id.* at 2:14-18.
[128] *See id.* at 2:14-20.
[129] *Id.* at 2:55-57.
[130] *Id.* at 2:65-67.

> computer executable code executable by the processor for connecting to a web service, the computer executable code comprising:
>
> code for selecting a web service;
>
> code for selecting a server among one or more servers capable of running the selected web service, the selected server being selected independent of input from a requesting application subsequent to selection of the web service;
>
> code for determining a real address for the selected web service running on the selected server; and
>
> code for connecting to the selected web service running on the selected server using the determined real address.

298. Claim 15 of the ’976 Patent reads as follows:

> The computer system of claim 14, wherein selecting a web service comprises searching a directory of web services.

299. Claim 18 of the ’976 Patent reads as follows:

> The computer system of claim 14, wherein selecting a server among one or more servers capable of running the selected web service comprises performing a distributed scheduling algorithm.

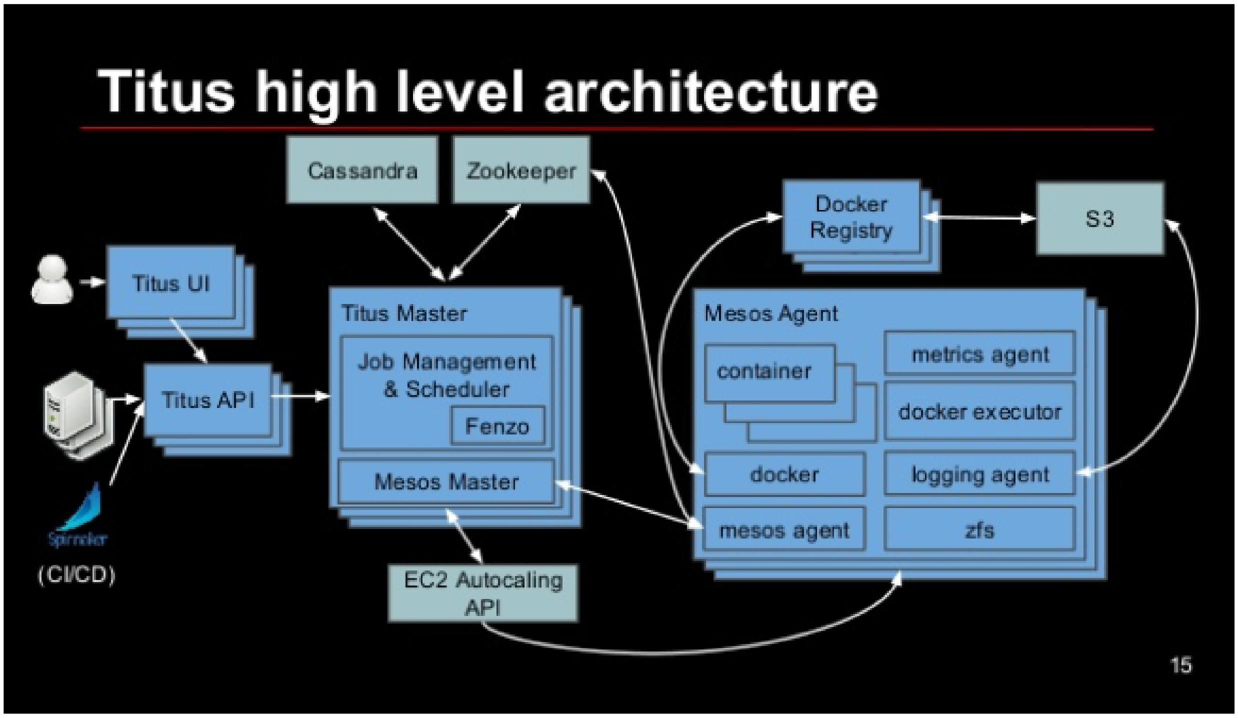
300. Claim 20 of the ’976 Patent reads as follows:

> The computer system of claim 18, wherein the distributed scheduling algorithm is a dynamic algorithm.

301. Thus, claims 14, 15, 19, and 20 are directed to a novel system for managing the provision of web services using a distributed computing system comprising multiple servers capable of running that web service. Upon information and belief, this system was not well-known, routine, or conventional at the time of the ’976 Patent.

302. Upon information and belief, Netflix directly infringes at least claims 14, 15, 18, and 20 of the ’976 Patent, at least in the exemplary manner described below.

303. Netflix directly infringes the ’976 Patent by making, using, offering to sell, and/or selling in the United States its Netflix service, which relies on the Titus/Fenzo container architecture (depicted below) to deploy and manage virtual computing resources in flexible, scalable manner.



Titus high level architecture

Source: https://www.slideshare.net/aspyker/netflix-and-containers-titus.

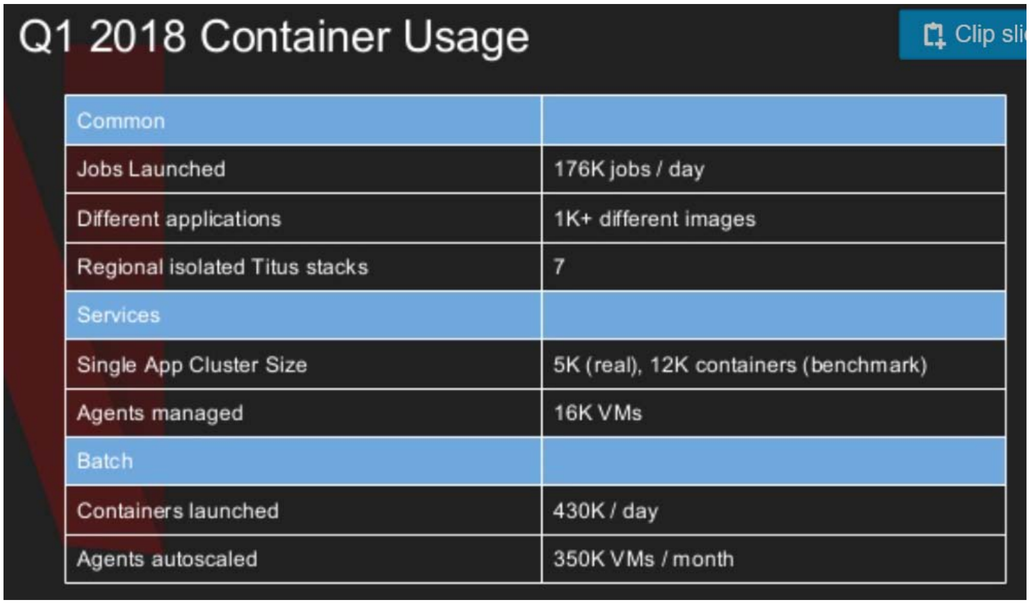
304. In the words of Netflix:

> Titus is a container management platform that provides scalable and reliable container execution and cloud-native integration with Amazon AWS. Titus was built internally at Netflix and is used in production to power Netflix streaming, recommendation, and content systems.[131]

[131] https://netflix.github.io/titus/.

305. According to Netflix, "Titus powers critical aspects of the Netflix business, from video streaming, recommendations, and machine learning, big data, content encoding, studio technology, internal engineering tools, and other Netflix workloads." [132]

306. As Netflix explains, "Titus offers a convenient model for managing compute resources."[133] On information and belief, Netflix currently uses the Titus platform to launch and manage millions of containers per week, and to host thousands of applications globally across tens of thousands of virtual machines.[134]



Q1 2018 Container Usage

Common	
Jobs Launched	176K jobs / day
Different applications	1K+ different images
Regional isolated Titus stacks	7
Services	
Single App Cluster Size	5K (real), 12K containers (benchmark)
Agents managed	16K VMs
Batch	
Containers launched	430K / day
Agents autoscaled	350K VMs / month

Source: https://www.slideshare.net/aspyker/container-world-2018.

307. With regard to claim 14 of the '976 Patent, Netflix practices "[a] computer system comprising: a processor; and a computer recording medium including computer executable code executable by the processor for connecting to a

[132] https://netflixtechblog.com/titus-the-netflix-container-management-platform-is-now-open-source-f868c9fb5436.

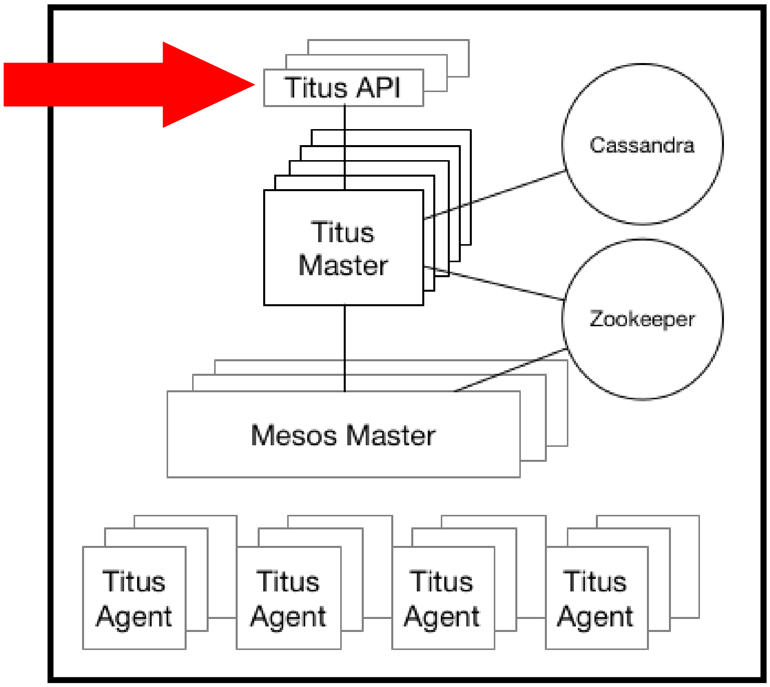
[133] https://medium.com/netflix-techblog/titus-the-netflix-container-management-platform-is-now-open-source-f868c9fb5436.

[134] *Id.*

web service…" For example, Netflix's Titus system allows users at Netflix to connect to the platform (a web service), which deploys and manages containers—virtual computing resources—to run the users' applications and otherwise accommodate their computing needs. On information and belief, the Titus software (computer executable code executable by the processor for connecting to a web service) runs on a plurality of servers in the cloud, where it is stored in the servers' computer recording medium (e.g., computer memory) and executed by the servers' processors.

308. The Netflix system also includes "code for selecting a web service." For instance, the Titus API, also referred to as the Titus Gateway (shown by the red arrow in the diagram below), handles requests from users at Netflix, allowing users to provide the details of the specific container management services they require.[135] As Netflix explains, "[w]ork in Titus is described by a job specification that details what to run (e.g., a container image and entry point), metadata (e.g., the job's purpose and who owns it), and what resources are required to run it, such as CPU, memory or scheduling constraints (e.g., availability zone balancing or host affinity)".[136]

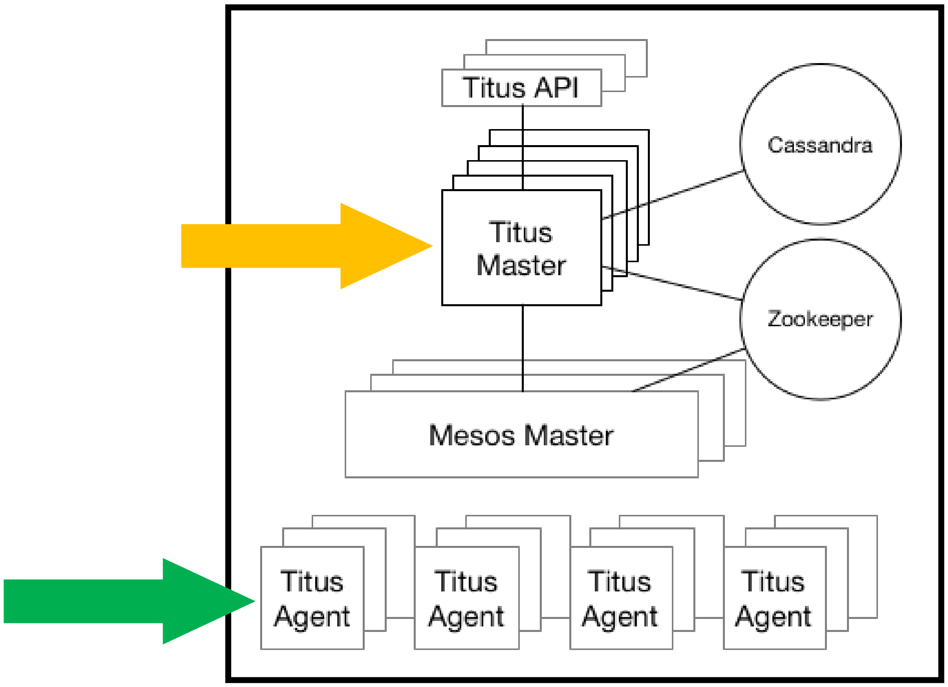
[135] *See* https://netflix.github.io/titus/overview/ ("The Titus Gateway is a scalable API tier that handles direct requests from clients and users.").
[136] *Id.*



Source: https://netflix.github.io/titus/overview/.

309. The Netflix system also includes "code for selecting a server among one or more servers capable of running the selected web service." For example, as Netflix explains, "Titus is run in production at Netflix, managing thousands of AWS EC2 instances,"[137] each of which "functions as a virtual private server."[138] The Titus Master (identified by the orange arrow in the diagram below) "is responsible for persisting job and task information, scheduling tasks, and managing the pool of EC2 Agents"[139] (each a Titus Agent, identified by the green arrow in the diagram below). As Netflix further explains, the Titus Master "schedules tasks onto the Agents with available resources and scales the pool of Titus Agents up or down in response to demand."[140]

[137] https://netflix.github.io/titus/
[138] https://en.wikipedia.org/wiki/Amazon_Elastic_Compute_Cloud
[139] https://netflix.github.io/titus/overview/
[140] *Id.*



Source: https://netflix.github.io/titus/overview/.

310. Specifically, the Titus Master utilizes a Netflix-developed software called Fenzo to select the server (i.e., the Titus Agent) to use in providing the service requested by the user. As Netflix explains:

> The Fenzo scheduler selects potential hosts to assign tasks to based on whether the hosts have sufficient resources to successfully execute the tasks. Each task has its own resource requirements, each host offer lists its available resources, and Fenzo maintains an accounting of which resources are available and which are already assigned on each host.[141]

311. Within this process, the "selected server" is selected "independent of input from a requesting application subsequent to selection of the web service." For example, as Netflix explains:

> One of the benefits of using containers through Titus is that it abstracts much of the machine-centric management that applications were doing in VMs. In many cases,

[141] https://github.com/Netflix/Fenzo/wiki/Introduction.

> users can tell Titus to "run this application" without worrying about where or on which instance type the container runs.[142]

312. The Netflix system includes "code for determining a real address for the selected web service running on the selected server." For example, Netflix engineers have explained that "[w]hen Titus is preparing to launch a new container, it creates a network namespace for it, assigns it a specific IP address from an ENI [an Elastic Network Interface], and connects the container's network namespace to the host's using a *veth* (virtual Ethernet) interface."[143]

313. Finally, the Netflix system includes "code for connecting to the selected web service running on the selected server using the determined real address." For example, on information and belief, the Titus system uses the IP address created for the containers (as described in previous paragraph) in utilizing and managing those containers.

314. With regard to claim 15 of the '976 Patent, within the Titus system, "selecting a web service comprises searching a directory of web services." For example, on information and belief, the Fenzo scheduler used by the Titus Master tracks and maintains data on the Titus Agents and their available resources, and assigns tasks to the Agents based on whether they have sufficient resources to successfully execute those tasks.[144]

315. With regard to claim 18 of the '976 Patent, within the Titus system, "selecting a server among one or more servers capable of running the selected web service comprises performing a distributed scheduling algorithm." For instance,

[142] https://queue.acm.org/detail.cfm?id=3158370.

[143] *Id.*

[144] https://github.com/Netflix/Fenzo/wiki/Introduction.

Netflix publications explain how the Fenzo scheduler utilized by the Titus Master "adopts a strategy of scoring hosts [i.e., Titus Agents] for a task placement, with higher scores indicating better 'fitness.'"[145] Fenzo selects the Titus Agent to which to assign tasks "based on whether the hosts have sufficient resources to successfully execute the tasks."[146]

316. With regard to claim 20 of the '976 Patent, the "distributed scheduling algorithm" described in the preceding paragraph "is a dynamic algorithm." For example, as explained in the preceding paragraph, the Fenzo scheduler takes into account the resources then available on the potential Titus Agents in determining whether the Agent is sufficiently "fit" to handle the task Fenzo seeks to assign.

317. Upon information and belief, Netflix directly infringes other claims of the '976 Patent as well, including for the reasons discussed in the preceding paragraphs.

318. At least as of the filing and service of this Amended Complaint, Netflix has had knowledge of the '976 Patent and Netflix's infringement thereof.

319. Netflix's infringement of the '976 Patent, which is knowing and willful at least as of the filing of this Amended Complaint, has caused and continues to cause damage to Avago. Avago is entitled to recover damages sustained as a result of Netflix's wrongful acts in an amount subject to proof at trial.

[145] *Id.*

[146] *Id.*

ELEVENTH CLAIM FOR RELIEF

(Infringement of U.S. Patent No. 7,457,722)

320. The Broadcom Entities reallege and incorporate by reference the allegations of paragraphs 1-319 set forth above.

321. The ’722 Patent, entitled “Correlation of Application Instance Life Cycle Events in Performance Monitoring,” was duly and legally issued on November 25, 2008 from a patent application filed on November 17, 2004, with Tomer Shain and John A. Colgrove as the named inventors. A copy of the ’722 Patent is attached hereto as **Exhibit K**.

322. The ’722 Patent was assigned to Avago, which currently holds all substantial rights, title, and interest in and to the ’722 Patent.

323. Pursuant to 35 U.S.C. § 282, the ’722 Patent is presumed valid.

324. The ’722 Patent is directed to an improvement in the functionality of complex distributed computer networks.

325. The ’722 Patent addresses specific technical challenges that arose in information technology and computational services as the service levels required by users continued to increase, while the computer networks used to provide those services became increasingly complex.

326. The ’722 Patent explains that:

> In the information technology (IT) departments of modern organizations, one of the biggest challenges is meeting the increasingly demanding service levels required by users. With more and more applications directly accessible to customers via automated interfaces such as the world wide web, “normal” business hours for many enterprises are now 24 hours a day, 7 days a week. As a result, the importance of monitoring and maintaining the quality of

computational services has increased dramatically.[147]

327. The ’722 Patent also describes the complexities of monitoring the distributed computing environments used to provide these increasingly demanding services. It explains how “[m]onitoring the server side may comprise monitoring one or more applications executing on a cluster of nodes.”[148] Clusters of nodes “may comprise a utility computing environment, wherein the computational power of one or more servers may be purchased as needed.”[149] “Alternatively, dozens or hundreds of nodes may be organized into interconnected tiers of web servers, application servers and databases.”[150] “Each node in such a system may execute multiple instances of one or more applications, with each instance operable to handle a different client request.”[151]

328. As the ’722 Patent explains, in these sorts of distributed computing environments, “application instances may be created or destroyed as demand changes.”[152] Application instances may also “migrate from node to node in response to a hardware or software failure, or in response to a load-balancing algorithm, for example.”[153]

329. Existing monitoring systems did not adequately address these challenges. The ’722 Patent describes, for instance, how “monitoring systems may be unable to track the migration of instances from node to node, thereby preventing the monitoring system from presenting a complete picture of a given application

[147] ’722 Patent, 1:12-20.
[148] *Id.* at 1:25-26.
[149] *Id.* at 1:27-29.
[150] *Id.* at 1:29-31.
[151] *Id.* at 1:32-34.
[152] *Id.* at 1:35-36.
[153] *Id.* at 1:37-39.

instance."[154] Additionally, "monitoring systems may not be operable to monitor data on the creation or destruction of application instances."[155] Further, the '722 Patent explains that then-existing software performance monitors did not "correlate performance data with events, such as the creation, migration or destruction of a particular application instance, and across all instances of a particular application, application group, and/or technology tier."[156]

330. The '722 Patent claims specific, novel ways to address these technical challenges with a performance monitoring solution for application instances that includes instance life cycle event monitoring and that correlates the performance data with instance life cycle events. The claims of the '722 Patent are directed to new, improved methods and systems implementing this performance monitoring solution.

331. The methods and systems described in the '722 Patent improve the functionality of a distributed computer system by better monitoring the performance of the applications running in this environment.

332. Claim 17 of the '722 Patent reads as follows:

> A performance monitoring system, comprising:
>
> a processor;
>
> a memory coupled to the processor and storing program instructions executable by said processor to implement:
>
>> collecting performance data for one or more application instances, wherein the performance data is associated with the performance of said one or more application instances, wherein each

[154] *Id.* at 1:39-43.
[155] *Id.* at 1:43-45.
[156] *Id.* at 1:45-49.

> application instance is a computer program executing on a computer system;
>
> detecting one or more instance life cycle events associated with said one or more application instances, wherein said one or more instance life cycle events comprise at least one of: the creation of at least one of said one or more application instances, the destruction of at least one of said one or more application instances, and the migration of at least one of said one or more application instances;
>
> correlating said performance data to said one or more instance life cycle events; and
>
> storing the correlated performance data.

333. Claim 18 of the ’722 Patent reads as follows:

> The performance monitoring system of claim 17, wherein the one or more application instances are instances of an application, and wherein said correlating said performance data to one or more instance life cycle events comprises correlating the one or more instance life cycle events to the performance of the application.

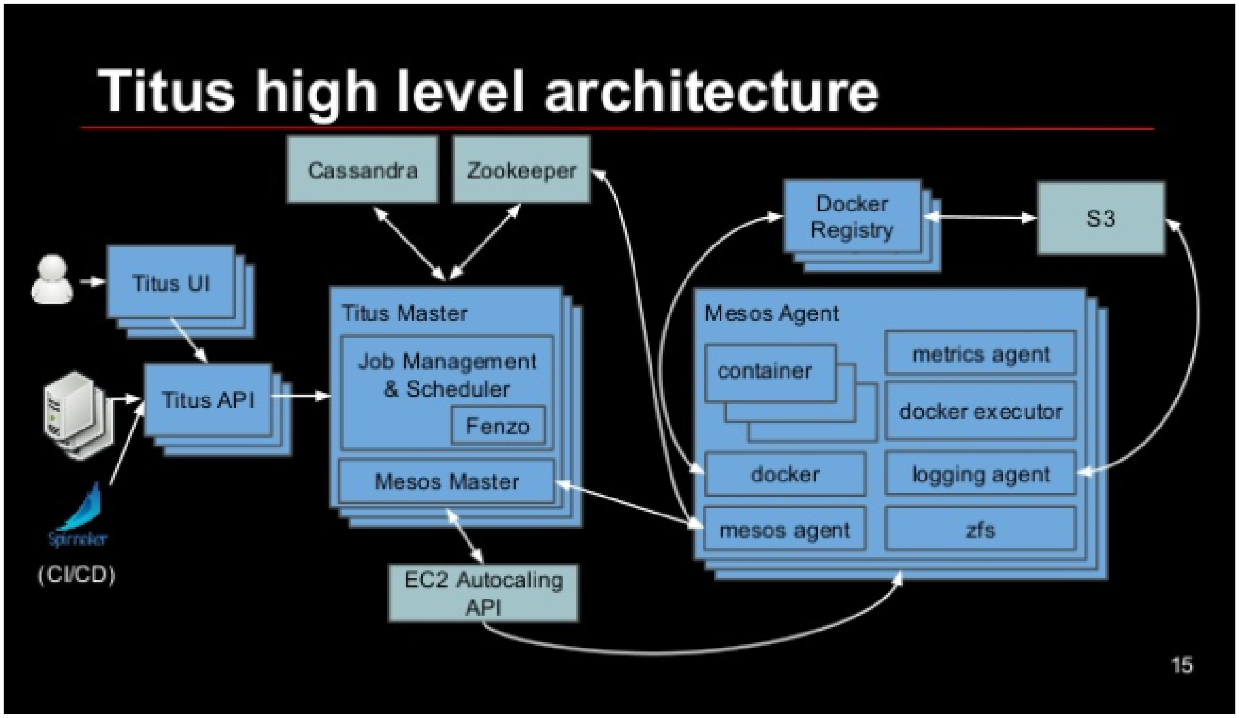
334. Claim 19 of the ’722 Patent reads as follows:

> The performance monitoring system of claim 18, wherein said correlating said performance data to one or more instance life cycle events comprises determining the change in performance of the application as the one or more application instances are created and destroyed.

335. Thus, claims 17, 18, and 19 are directed to a novel performance monitoring system for a distributed computing network that includes application instance life cycle event monitoring and that correlates the performance data with instance life cycle events, amongst other things. Upon information and belief, this system was not well-known, routine, or conventional at the time of the ’976 Patent.

336. Upon information and belief, Netflix directly infringes at least claims 17, 18, and 19 of the ’722 Patent, at least in the exemplary manner described below.

337. Netflix directly infringes the ’722 Patent by making, using, offering to sell, and/or selling in the United States its Netflix service, which relies on the Titus/Atlas architecture (depicted below) to deploy, monitor, and manage virtual computing resources in a flexible, efficient manner.



Titus high level architecture

Source: https://www.slideshare.net/aspyker/netflix-and-containers-titus.

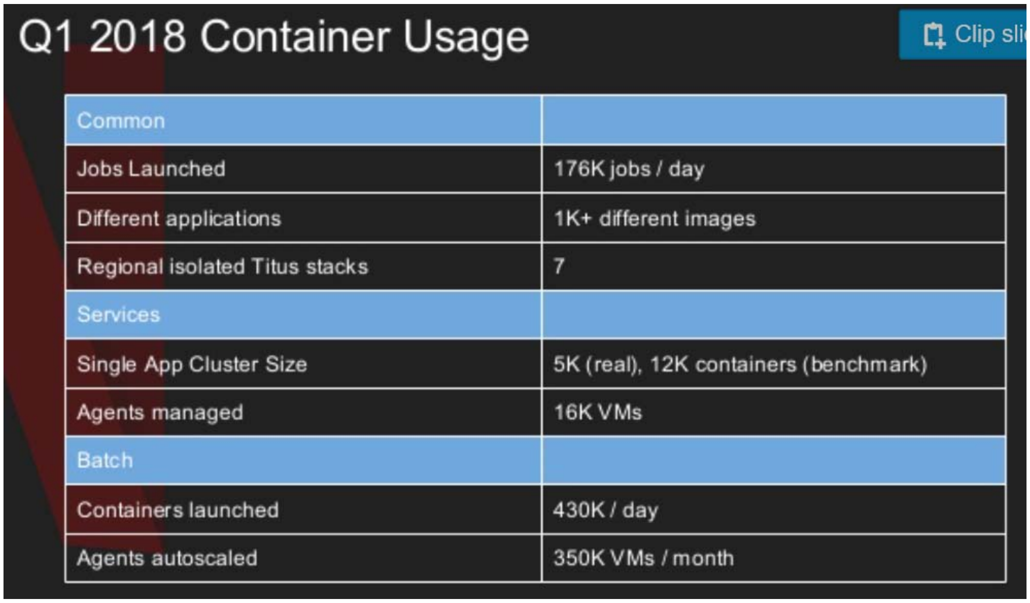
338. In the words of Netflix:

> Titus is a container management platform that provides scalable and reliable container execution and cloud-native integration with Amazon AWS. Titus was built internally at Netflix and is used in production to power Netflix streaming, recommendation, and content systems.[157]

[157] https://netflix.github.io/titus/.

339. According to Netflix, "Titus powers critical aspects of the Netflix business, from video streaming, recommendations, and machine learning, big data, content encoding, studio technology, internal engineering tools, and other Netflix workloads."[158]

340. On information and belief, Netflix currently uses the Titus platform to launch and manage millions of containers per week, and to host thousands of applications globally across tens of thousands of virtual machines.[159]



Q1 2018 Container Usage

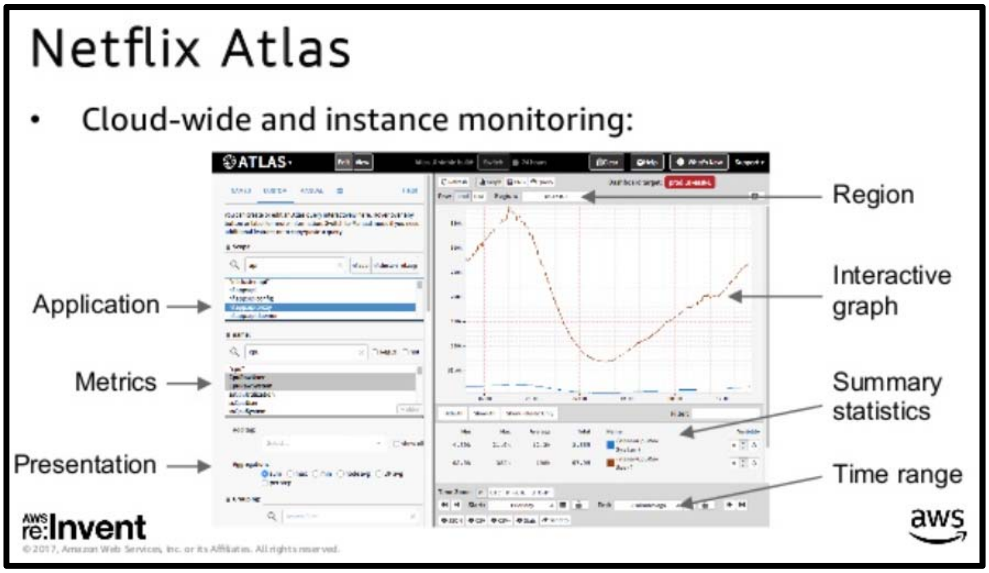
Common	
Jobs Launched	176K jobs / day
Different applications	1K+ different images
Regional isolated Titus stacks	7
Services	
Single App Cluster Size	5K (real), 12K containers (benchmark)
Agents managed	16K VMs
Batch	
Containers launched	430K / day
Agents autoscaled	350K VMs / month

Source: https://www.slideshare.net/aspyker/container-world-2018.

341. The Titus platform relies on Altas, a performance monitoring tool developed by Netflix, to monitor performance within the Titus system.

[158] https://netflixtechblog.com/titus-the-netflix-container-management-platform-is-now-open-source-f868c9fb5436.

[159] *See* https://medium.com/netflix-techblog/titus-the-netflix-container-management-platform-is-now-open-source-f868c9fb5436.



Netflix Atlas

- Cloud-wide and instance monitoring:

Application

Metrics

Presentation

Region

Interactive graph

Summary statistics

Time range

AWS re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

Source: https://www.slideshare.net/brendangregg/how-netflix-tunes-ec2-instances-for-performance.

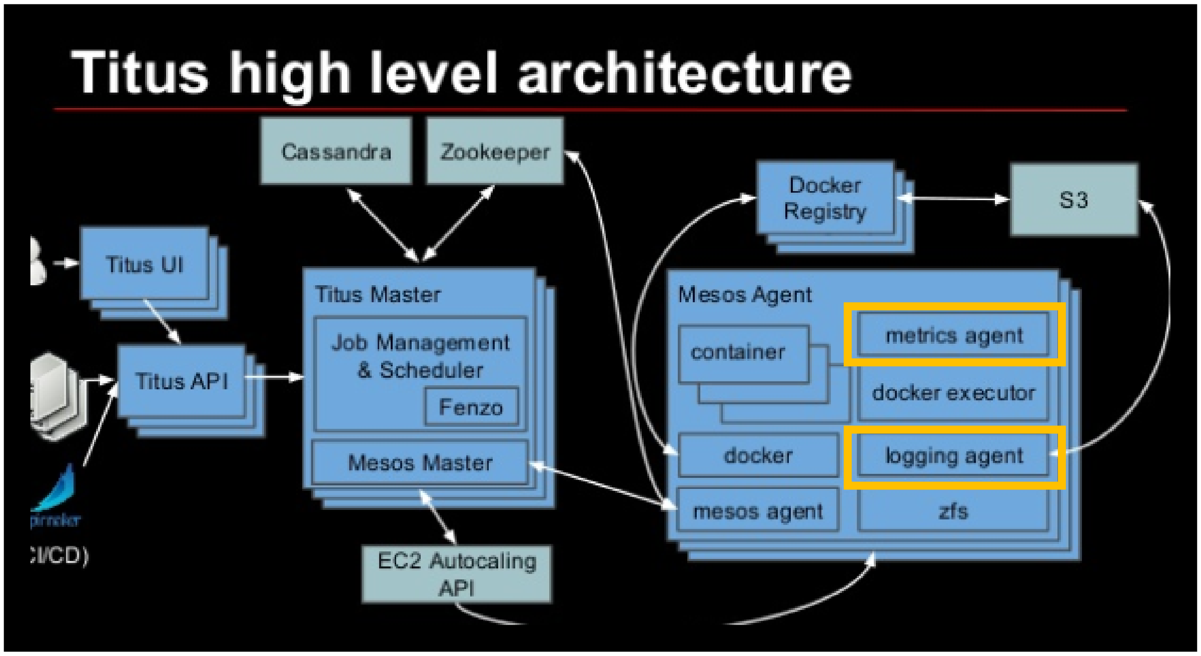
342. In Netflix's words, "Atlas was developed by Netflix to manage dimensional time series data for near real-time operational insight."[160] "Atlas captures operational intelligence . . . operational intelligence provides a picture of what is currently happening within a system."[161]

343. Netflix's Titus system utilizes an Atlas "agent" on each Titus Agent (depicted in yellow in the figure below) that provides "container-level system metrics (e.g., CPU and memory usage) from Titus agents."[162]

[160] *See* https://github.com/Netflix/atlas/wiki.

[161] *Id.*

[162] *See* https://queue.acm.org/detail.cfm?id=3158370.

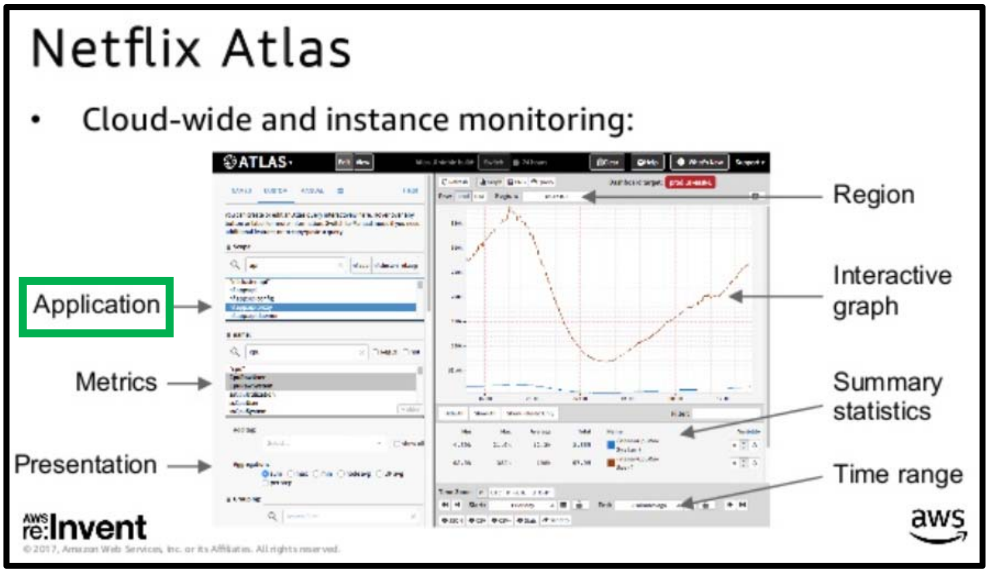


Source: https://www.slideshare.net/aspyker/netflix-and-containers-titus (orange annotations added)

344. With regard to claim 17 of the '722 Patent, Netflix practices a "performance monitoring system, comprising a processor" and "a memory coupled to the processor and storing program instructions executable by said processor." For example, Netflix's Titus system utilizes Atlas (a performance monitoring system). On information and belief, the Atlas software (instructions executable by a processor) runs on one or more servers in the cloud, where it is stored in the servers' memory and executable (and executed) by the servers' processors.

345. Netflix's system practices the step of "collecting performance data for one or more application instances, wherein the performance data is associated with the performance of said one or more application instances, wherein each application instance is a computer program executing on a computer system." For example, Netflix's Titus system integrates with Atlas to collect performance data on each

Titus Agent at the container level.[163] Each container on the Titus Agent represents an application instance and is a computer program executing on a computer system.



Netflix Atlas

- Cloud-wide and instance monitoring:

Source: https://www.slideshare.net/brendangregg/how-netflix-tunes-ec2-instances-for-performance (green annotation added).

346. Netflix's system also practices the step of "detecting one or more instance life cycle events associated with said one or more application instances, wherein said one or more instance life cycle events comprise at least one of: the creation of at least one of said one or more application instances, the destruction of at least one of said one or more application instances, and the migration of at least one of said one or more application instances." For instance, containers (i.e., "application instances") in Linux-based systems such as Titus rely on control

[163] *See, e.g.*, https://queue.acm.org/detail.cfm?id=3158370 ("Similarly, the health-check polling system was changed to query Titus and provide health-check polling for containers in addition to VMs. The on-instance Atlas telemetry agent was changed to collect and emit container-level system metrics (e.g., CPU and memory usage) from Titus agents. Previously, it collected only metrics for the entire host."); https://netflix.github.io/titus/overview/ ("The [Titus] Agent sets up on host resources, such as storage and networking resources, and launches the containers using Docker. The Agent monitors the task, reporting status and cleaning up resources when it complies.").

groups (or "cgroups") to track container processes and expose metrics about CPU, memory, and disk usage.[164] Among the metrics that may be retrieved are several "'counters', or values that can only go up, because they represent occurrences of a specific event."[165] Such events may include, for example, when a container is created or destroyed. For example, the container-level metric "cpuacct.stat" indicates the "amount of time a process has direct control of the CPU, executing process code" (i.e., "user" time) as well as the "time the kernel is executing system calls on behalf of the process" (i.e., "system" time).[166] In other words, the "cpuacct.stat" metric detects when a new container is launched and measures the amount of time that a process has been running on that container.

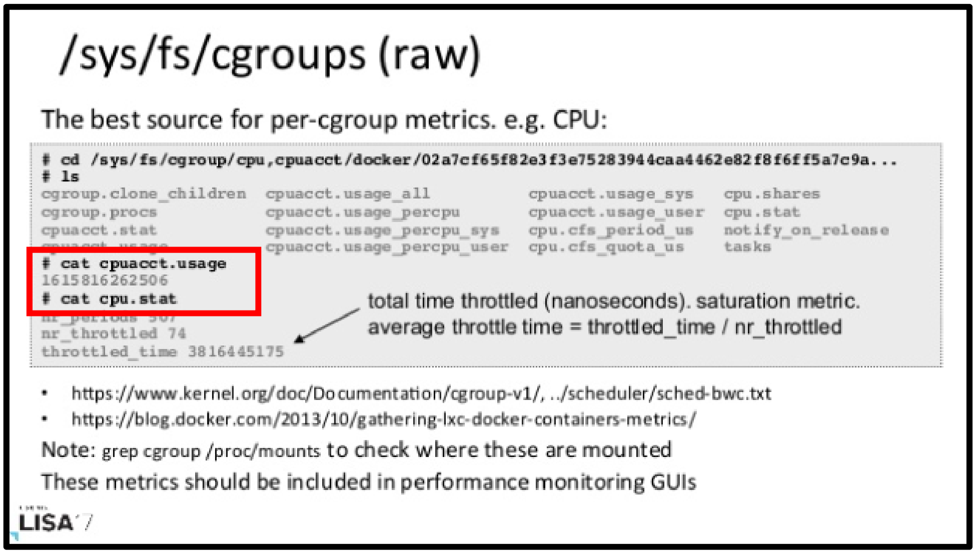
As another example, cgroups can also be used to detect how much CPU, memory, etc., a container (an "application instance") has used when a container exits or is terminated, how much time has elapsed since the container's launch (i.e., container uptime), and various other metrics.[167] As Netflix explains, the Titus/Atlas system utilizes at least the "cpuacct.stat" and container uptime metrics to detect at least when a container is created or destroyed.

[164] *See* https://docs.docker.com/config/containers/runmetrics/.

[165] *Id.*

[166] *Id.*

[167] *Id.*; *see also* https://docs.docker.com/engine/reference/commandline/ps/ (explaining how the "docker ps" command can be used in Docker-based container systems such as Titus to detect when a container is created or terminated).



/sys/fs/cgroups (raw)

The best source for per-cgroup metrics. e.g. CPU:

```
# cd /sys/fs/cgroup/cpu,cpuacct/docker/02a7cf65f82e3f3e75283944caa4462e82f8f6ff5a7c9a...
# ls
cgroup.clone_children  cpuacct.usage_all          cpuacct.usage_sys   cpu.shares
cgroup.procs           cpuacct.usage_percpu       cpuacct.usage_user  cpu.stat
cpuacct.stat           cpuacct.usage_percpu_sys   cpu.cfs_period_us   notify_on_release
cpuacct.usage          cpuacct.usage_percpu_user  cpu.cfs_quota_us    tasks
# cat cpuacct.usage
1615816262506
# cat cpu.stat
nr_periods 507
nr_throttled 74
throttled_time 3816445175
```

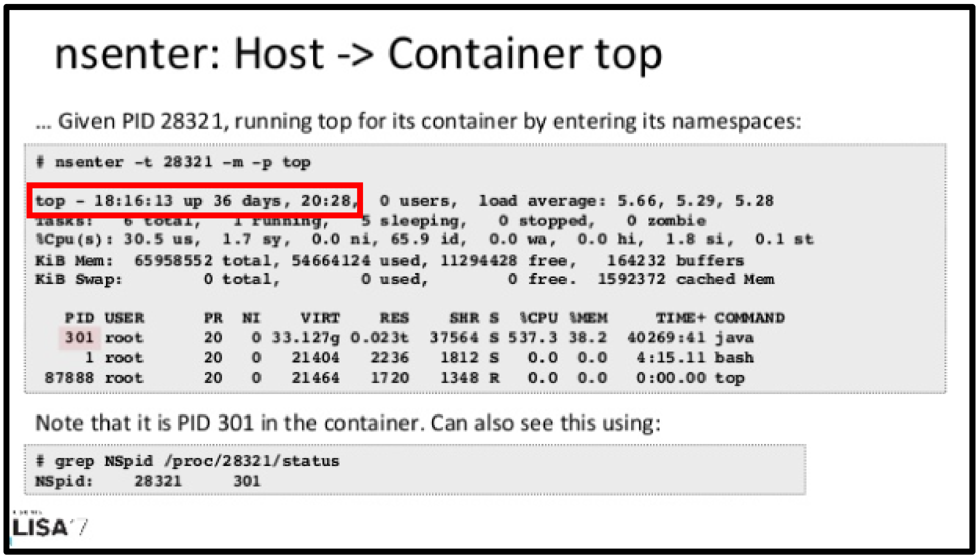
total time throttled (nanoseconds). saturation metric.
average throttle time = throttled_time / nr_throttled

- https://www.kernel.org/doc/Documentation/cgroup-v1/, ../scheduler/sched-bwc.txt
- https://blog.docker.com/2013/10/gathering-lxc-docker-containers-metrics/

Note: grep cgroup /proc/mounts to check where these are mounted

These metrics should be included in performance monitoring GUIs

LISA17



nsenter: Host -> Container top

... Given PID 28321, running top for its container by entering its namespaces:

```
# nsenter -t 28321 -m -p top

top - 18:16:13 up 36 days, 20:28,  0 users,  load average: 5.66, 5.29, 5.28
Tasks:   6 total,   1 running,   5 sleeping,   0 stopped,   0 zombie
%Cpu(s): 30.5 us,  1.7 sy,  0.0 ni, 65.9 id,  0.0 wa,  0.0 hi,  1.8 si,  0.1 st
KiB Mem:  65958552 total, 54664124 used, 11294428 free,   164232 buffers
KiB Swap:        0 total,        0 used,        0 free.  1592372 cached Mem

   PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
   301 root      20   0 33.127g 0.023t  37564 S 537.3 38.2  40269:41 java
     1 root      20   0   21404   2236   1812 S   0.0  0.0   4:15.11 bash
 87888 root      20   0   21464   1720   1348 R   0.0  0.0   0:00.00 top
```

Note that it is PID 301 in the container. Can also see this using:

```
# grep NSpid /proc/28321/status
NSpid:    28321     301
```

LISA17

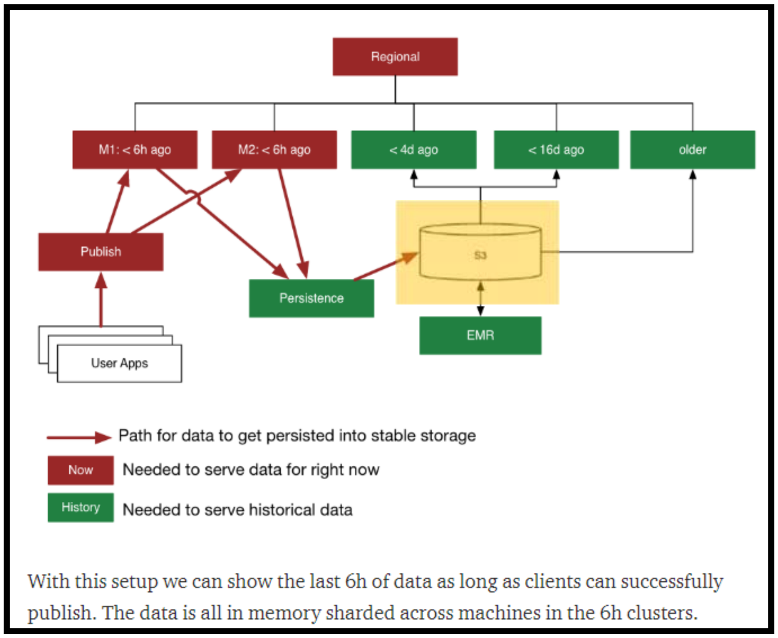
Source: https://www.slideshare.net/brendangregg/lisa17-container-performance-analysis (red annotations added).

347. Netflix's system also practices the step of "correlating said performance data to said one or more instance life cycle events." For example, as Netflix explains, "Atlas is a backend for storing and querying dimensional time series data. . . . In Atlas[,] each time series is paired with metadata called tags that allow us to query and group the data."[168] Thus, the Atlas system correlates the

[168] *See* https://github.com/Netflix/atlas/wiki/Concepts.

container data collected from Titus to visualize the change in performance over time, including the performance of the system before and after the creation and destruction of individual containers.

348. Finally, the Netflix system practices the step of "storing the correlated performance data." For example, Netflix explains that "Atlas features in-memory data storage, allowing it to gather and report very large numbers of metrics, very quickly."[169] Netflix further explains that this data is stored in one or more S3 "cloud" servers controlled by Netflix.


Regional
M1: < 6h ago
M2: < 6h ago
< 4d ago
< 16d ago
older
Publish
Persistence
S3
EMR
User Apps
Path for data to get persisted into stable storage
Now
Needed to serve data for right now
History
Needed to serve historical data

With this setup we can show the last 6h of data as long as clients can successfully publish. The data is all in memory sharded across machines in the 6h clusters.

Source: https://netflixtechblog.com/introducing-atlas-netflixs-primary-telemetry-platform-bd31f4d8ed9a (orange annotation added).

349. With regard to claim 18 of the '722 Patent, Netflix practices "[t]he performance monitoring system of claim 17, wherein the one or more application instances are instances of an application." For example, as previously discussed, each container on the Titus Agent represents an application instance and is a

[169] *See* https://github.com/Netflix/atlas/wiki.

computer program executing on a computer system. As Netflix explains, "[u]sing containers, Titus runs any batch application letting the user specify exactly what application code and dependencies are needed."[170] "Container images [i.e., instances] provide an easy way to build application-specific images that have only what the application needs."[171]

350. The Netflix system practices the system of claim 18, "wherein said correlating said performance data to one or more instance life cycle events comprises correlating the one or more instance life cycle events to performance of the application." For example, as previously discussed, Netflix uses the "Atlas telemetry agent . . . to collect and emit container-level system metrics (e.g., CPU and memory usage) from Titus agents."[172] As Netflix explains, these container-level metrics include such metrics as "cpuacct.stat" and container uptime (using, for example, the "docker ps" command) that detect and emit data in response to the occurrence of specific events such as the creation or destruction of a container.[173] As previously explained, Atlas correlates these container-level metrics by enabling users to compare the change in a container's performance over time.[174]

351. With regard to claim 19 of the '722 Patent, Netflix practices "[t]he performance monitoring system of claim 18, wherein said correlating said performance data to one or more instance life cycle events comprises determining the change in performance of the application as the one or more application

[170] https://netflixtechblog.com/the-evolution-of-container-usage-at-netflix-3abfc096781b.

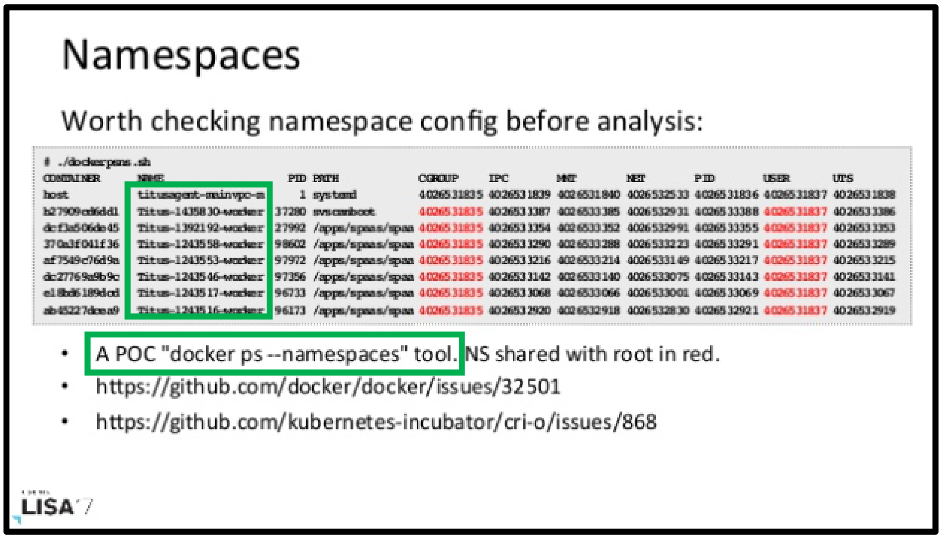
[171] https://queue.acm.org/detail.cfm?id=3158370.

[172] *Id.*

[173] *See* https://www.slideshare.net/brendangregg/lisa17-container-performance-analysis; *see also*, https://docs.docker.com/engine/reference/commandline/ps/.

[174] *See* https://github.com/Netflix/atlas/wiki/Concepts.

instances are created and destroyed." For example, as Netflix explains, the Atlas telemetry agent collects container-level metrics such as "cpuacct.stat" and container uptime (using, for example, the "docker ps" command) that detect and emit data in response to the occurrence of specific events such as the creation or destruction of a container.[175] As Netflix further explains, Atlas displays these metrics along with various other performance data in a time series format that determines the change in performance of an individual container or group of containers over a period of time.[176]



Namespaces

Worth checking namespace config before analysis:

```
# ./dockerpsns.sh
CONTAINER     NAME                  PID PATH             CGROUP     IPC        MNT        NET        PID        USER       UTS
host          titusagent-mainvpc-m    1 systemd          4026531835 4026531839 4026531840 4026532533 4026531836 4026531837 4026531838
b27909cd6dd1  Titus-1435830-worker 37280 svscanboot      4026531835 4026533387 4026533385 4026532931 4026533388 4026531837 4026533386
dcf3e506de45  Titus-1392192-worker 27992 /apps/spaas/spaa 4026531835 4026533354 4026533352 4026532991 4026533355 4026531837 4026533353
370a3f041f36  Titus-1243558-worker 98602 /apps/spaas/spaa 4026531835 4026533290 4026533288 4026533223 4026533291 4026531837 4026533289
af7549c76d9a  Titus-1243553-worker 97972 /apps/spaas/spaa 4026531835 4026533216 4026533214 4026533149 4026533217 4026531837 4026533215
dc27769a9b9c  Titus-1243546-worker 97356 /apps/spaas/spaa 4026531835 4026533142 4026533140 4026533075 4026533143 4026531837 4026533141
e18bd6189dcd  Titus-1243517-worker 96733 /apps/spaas/spaa 4026531835 4026533068 4026533066 4026533001 4026533069 4026531837 4026533067
ab45227dcea9  Titus-1243516-worker 96173 /apps/spaas/spaa 4026531835 4026532920 4026532918 4026532830 4026532921 4026531837 4026532919
```

- A POC "docker ps --namespaces" tool. NS shared with root in red.
- https://github.com/docker/docker/issues/32501
- https://github.com/kubernetes-incubator/cri-o/issues/868

Source: https://www.slideshare.net/brendangregg/lisa17-container-performance-analysis (green annotations added).

352. Upon information and belief, Netflix directly infringes other claims of the '722 Patent as well, including for the reasons discussed in the preceding paragraphs.

[175] *See* https://www.slideshare.net/brendangregg/lisa17-container-performance-analysis; *see also*, https://docs.docker.com/engine/reference/commandline/ps/ (explaining how to use the "docker ps" command to detect when a container is created or destroyed).

[176] *See* https://github.com/Netflix/atlas/wiki/Concepts.

353. At least as of the filing and service of this Amended Complaint, Netflix has had knowledge of the '722 Patent and Netflix's infringement thereof.

354. Netflix's infringement of the '722 Patent, which is knowing and willful at least as of the filing of this Amended Complaint, has caused and continues to cause damage to Avago. Avago is entitled to recover damages sustained as a result of Netflix's wrongful acts in an amount subject to proof at trial.

TWELFTH CLAIM FOR RELIEF

(Infringement of U.S. Patent No. 8,365,183)

355. The Broadcom Entities reallege and incorporate by reference the allegations of paragraphs 1-354 set forth above.

356. The '183 Patent, entitled "System and Method for Dynamic Resource Provisioning for Job Placement," was duly and legally issued on January 29, 2013 from a patent application filed on September 2, 2008, with Kouros Esfahany, Rich Lau, and Michael Chiaramonte as the named inventors. A copy of the '183 Patent is attached hereto as **Exhibit L**.

357. The '183 Patent was assigned to Avago, which currently holds all substantial rights, title, and interest in and to the '183 Patent.

358. Pursuant to 35 U.S.C. § 282, the '183 Patent is presumed valid.

359. The '183 Patent is directed to an improvement in the functionality of complex distributed computing systems.

360. The '183 Patent addresses specific technical challenges that arose in distributed computing systems, particularly in large groups of computer systems

where multiple users require computing resources to perform various processing activities.[177]

361. The ’183 Patent specification explains that for distributed computing systems that share resources across multiple users, the various processing activities submitted by users may compete with one another for system resources, resulting in poor performance or system failure by the networked computers.[178] Further, while a distributed system may have several parallel computer devices (e.g., servers) available to perform the job, not every one of the available servers may be appropriate or even suitable for performing the job.[179] “For example, some of the available computer devices may not be adequately provisioned to perform the job while others may be too busy to perform the job in spite of being adequately provisioned.”[180] This too may result in poor performance or system failures within the distributed computing system.

362. As the ’183 Patent specification explains, one solution addressing this problem is to form complex rule statements using, for example, “an overall utilization value” (e.g., a calculated aggregate of performance metrics that may be based on user-defined parameters), “individually collected utilization metrics” (e.g., performance metrics for individual devices in the distributed system), and “user-defined parameters” in order to “identify the best computer device available to perform a job, to make decisions about when it is appropriate to take user defined actions, or to provide a new computer device for new work to be performed.”[181]

[177] ’183 Patent, 1:13-20.
[178] *Id.*
[179] *Id.* at 2:33-41.
[180] *Id.*
[181] *Id.* at 2:58-3:2.

363. The invention of the ’183 Patent enables distributed computing systems to make real-time intelligent decisions about the state of user defined computing resource pools while increasing the reliability and efficiency of the system.[182] As the specification explains, the ability to dynamically provision and place tasks within a distributed system is especially beneficial in the context of data centers that may have hundreds or even thousands of computer devices from which to choose.[183]

364. Yet another advantage of the ’183 Patent’s invention is that it “segregate[s] server level rules from service level resource utilization,” by creating a separate management structure that is flexible, reactive, and can readily adapt to meet the demands of virtually any distributed computing system regardless of scale.[184]

365. The ’183 Patent claims specific, novel ways to address these technical challenges and achieve at least the advantages described above using unconventional methods, systems, and apparatuses that enable the reliable, efficient, and dynamic provision of resources in a distributed computer system.

366. The methods, systems, and apparatuses described in the ’183 Patent improve the functionality of a distributed computer system by enabling the reliable, efficient, and dynamic provision of its computing resources to better handle computing tasks.

367. Independent claim 11 of the ’183 Patent reads as follows:

> A system for dynamic resource provisioning for job

[182] *Id.* at 2:53-58.
[183] *Id.* at 2:46-49.
[184] *See id.* at 2:7-9.

> placement, comprising:
>
> an interface operable to receive a request to perform a job on an unspecified computer device from a client machine;
>
> one or more processors operable to:
>
> determine one or more job criteria for performing the job, the one or more job criteria defining one or more operational characteristics needed for a computer device to perform the job;
>
> determine one or more utilization criteria for performing the job;
>
> provide a list of available computer devices, the list comprising a plurality of computer devices currently provisioned to perform computer operations;
>
> from the list of available computer devices, determine a list of suitable computer devices for performing the job by comparing operational characteristics for each available computer device with the job criteria, the list of suitable computer devices comprising one or more computer devices having operational characteristics that satisfy the job criteria;
>
> use the utilization criteria to determine whether one or more underutilized computer devices exist on the list of suitable computer devices, the one or more underutilized computer devices having a suitable level of utilization for performing the job; and
>
> if the one or more underutilized computer devices exist, identify the one or more underutilized computer devices to the client machine.

368. Dependent claim 12 of the ’183 Patent reads as follows:

> The system of claim 11, wherein the one or more processors are operable to use the utilization criteria to identify one or more underutilized computer devices by, for each computer device on the list of suitable computer devices:

> determining a utilization value;
>
> comparing the utilization value with the utilization criteria; and
>
> removing the each device from the list of suitable computer devices if the utilization value does not satisfy the utilization criteria.

369. Dependent claim 15 of the ’183 Patent reads as follows:

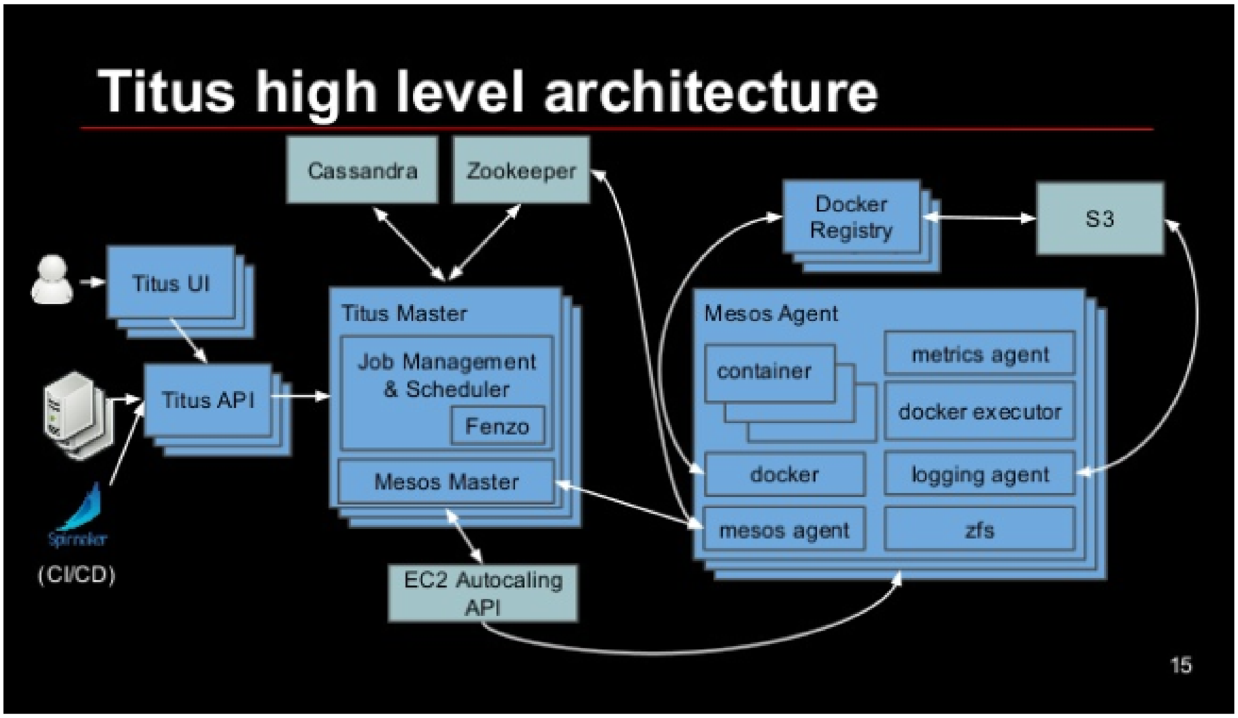
> The method of claim 11, wherein the one or more processors are configured to:
>
> use the utilization criteria to determine whether one or more underutilized computer devices exist on the list of suitable computer devices by identifying a computer device having a lowest level of activity relative to other computer devices on the list of suitable computer devices; and
>
> identify the computer device having the lowest level of utilization to the client machine.

370. Thus, claims 11, 12, and 15 are directed to a novel system for the dynamic provision of computing resources within a distributed computing system that, amongst other things, utilizes job criteria, operational characteristics of the computing resources, and utilization criteria of the computing resources to place jobs without requiring the user to specify the computer device to be used. Upon information and belief, this system was not well-known, routine, or conventional at the time of the ’183 Patent.

371. Upon information and belief, Netflix directly infringes at least claims 11, 12, and 15 of the ’183 Patent, at least in the exemplary manner described below.

372. Netflix directly infringes the ’183 Patent by making, using, offering to sell, and/or selling in the United States its Netflix service, which relies on the

Titus/Fenzo container architecture (depicted below) to deploy and manage virtual computing resources in flexible, scalable manner.



Titus high level architecture

Source: https://www.slideshare.net/aspyker/netflix-and-containers-titus.

373. In the words of Netflix:

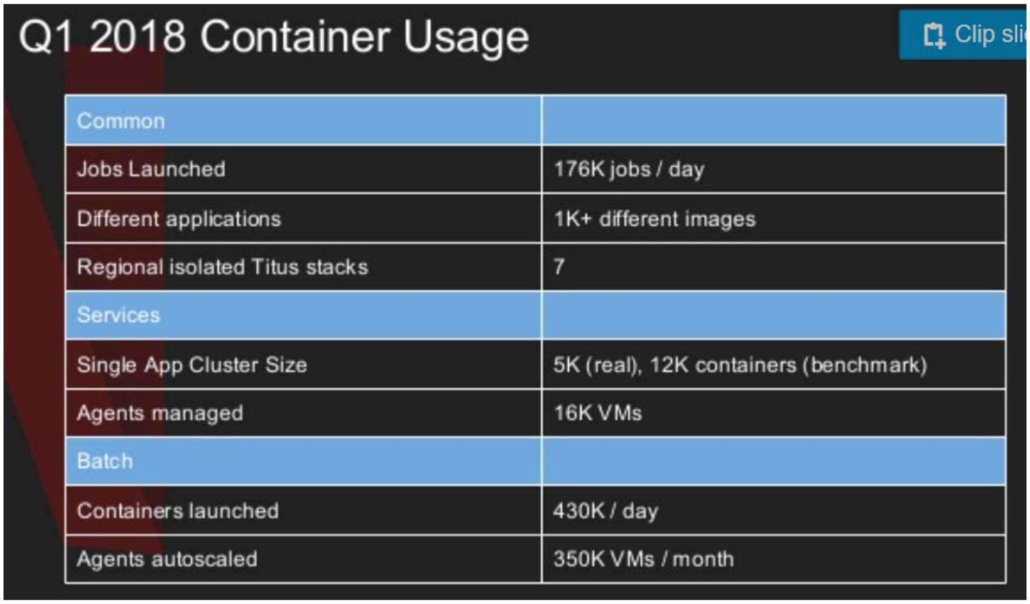
> Titus is a container management platform that provides scalable and reliable container execution and cloud-native integration with Amazon AWS. Titus was built internally at Netflix and is used in production to power Netflix streaming, recommendation, and content systems.[185]

374. According to Netflix, "Titus powers critical aspects of the Netflix business, from video streaming, recommendations, and machine learning, big data, content encoding, studio technology, internal engineering tools, and other Netflix workloads."[186]

[185] *See* https://netflix.github.io/titus/.

[186] https://netflixtechblog.com/titus-the-netflix-container-management-platform-is-now-open-source-f868c9fb5436.

375. As Netflix explains, "Titus offers a convenient model for managing compute [sic] resources."[187] On information and belief, Netflix currently uses the Titus platform to launch and manage millions of containers per week, and to host thousands of applications globally across tens of thousands of virtual machines.[188]



Q1 2018 Container Usage

Common	
Jobs Launched	176K jobs / day
Different applications	1K+ different images
Regional isolated Titus stacks	7
Services	
Single App Cluster Size	5K (real), 12K containers (benchmark)
Agents managed	16K VMs
Batch	
Containers launched	430K / day
Agents autoscaled	350K VMs / month

Source: https://www.slideshare.net/aspyker/container-world-2018.

376. According to Netflix, "Titus is a framework [that runs] on top of Apache Mesos, a cluster-management system that brokers available resources across a fleet of machines."[189] As Netflix explains, "Titus consists of a replicated, leader-elected scheduler called Titus Master."[190] The Titus Master is responsible for persisting job and task information, scheduling tasks, and managing the pool of EC2 virtual machine instances ("Titus Agents").[191]

[187] *See* https://medium.com/netflix-techblog/titus-the-netflix-container-management-platform-is-now-open-source-f868c9fb5436.

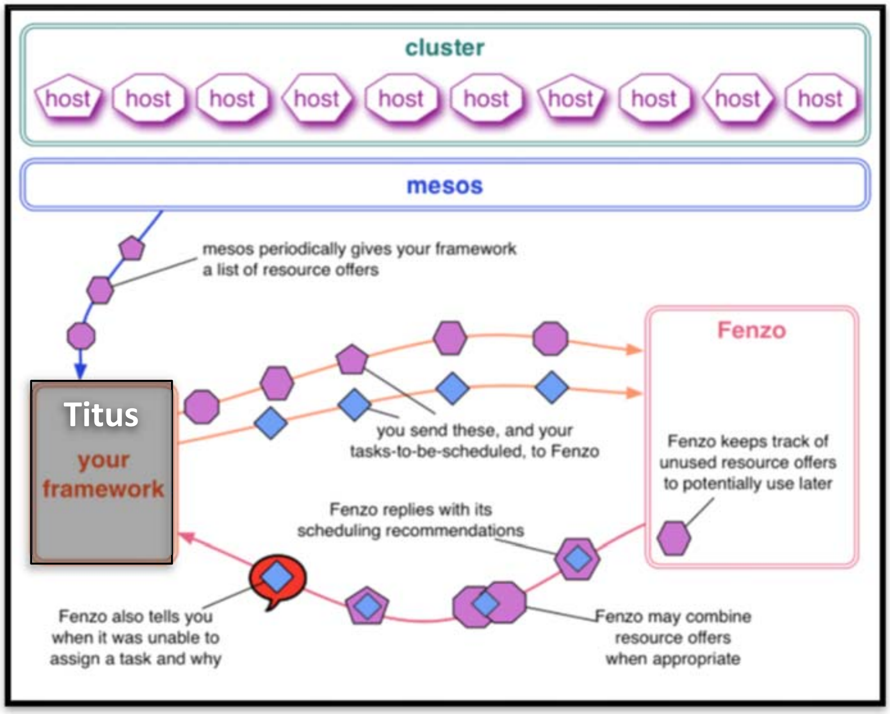
[188] *Id.*

[189] *Id.*

[190] *Id.*

[191] *Id.*

377. To perform these task placement and resource scheduling functions, the Titus Master relies on Fenzo, an extensible scheduler library developed and used internally by Netflix.[192] As Netflix explains, "Fenzo is a Java library that implements a generic task scheduler for Apache Mesos frameworks."[193] Apache Mesos is an open source project for managing clusters of computer devices (individually referred to as "hosts").[194]

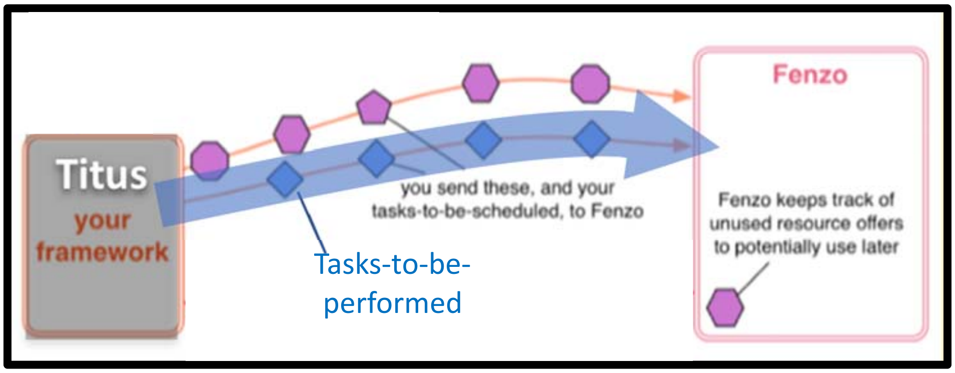

cluster
host
host
host
host
host
host
host
host
host
host
mesos
mesos periodically gives your framework a list of resource offers
Titus
your framework
you send these, and your tasks-to-be-scheduled, to Fenzo
Fenzo
Fenzo keeps track of unused resource offers to potentially use later
Fenzo replies with its scheduling recommendations
Fenzo also tells you when it was unable to assign a task and why
Fenzo may combine resource offers when appropriate

Source: https://github.com/Netflix/Fenzo/wiki (gray annotation added) .

378. With regard to claim 11, Netflix practices a "system for dynamic resource provisioning for job placement." For example, the Titus Master uses the Fenzo task scheduler "in conjunction with its agent autoscaling capabilities to grow and shrink the agent pool dynamically as workload demands."[195]

[192] *See* https://queue.acm.org/detail.cfm?id=3158370.
[193] *See* https://github.com/Netflix/Fenzo/wiki.
[194] *See* http://mesos.apache.org/.
[195] *See* https://queue.acm.org/detail.cfm?id=3158370.

379. Netflix also practices the step of "an interface operable to receive a request to perform a job on an unspecified computer device from a client machine." For example, the Titus Master provides an "interface" that receives requests to perform a job on one or more Titus Agents (unspecified computer device) from one or more Gateway instances (client machine). As Netflix explains, "[t]he Master receives requests from Gateway instances and creates and persists job and task info in response."[196] This request does not specify a computer device for performing a particular job—rather, the Titus/Fenzo architecture "consider[s] only the task's required resources and scheduling constraints" for task placement.[197]


Titus
your framework
you send these, and your tasks-to-be-scheduled, to Fenzo
Tasks-to-be-performed
Fenzo
Fenzo keeps track of unused resource offers to potentially use later

Source: https://github.com/Netflix/Fenzo/wiki (annotations added).

380. Netflix also practices the step of "one or more processors." For example, on information and belief, the Titus and Fenzo software run on one or more servers in the cloud, where the processors in those servers execute the software.

381. Netflix also practices the step wherein the one or more processors are operable to "determine one or more job criteria for performing the job, the one of more job criteria defining one or more operational characteristics needed for a

[196] *See* https://netflix.github.io/titus/overview/.
[197] *See* https://queue.acm.org/detail.cfm?id=3158370.

computer device to perform the job." For example, each task in Titus is assigned a job specification that defines the job criteria, including specific operational characteristics required for the task. In Netflix's words:

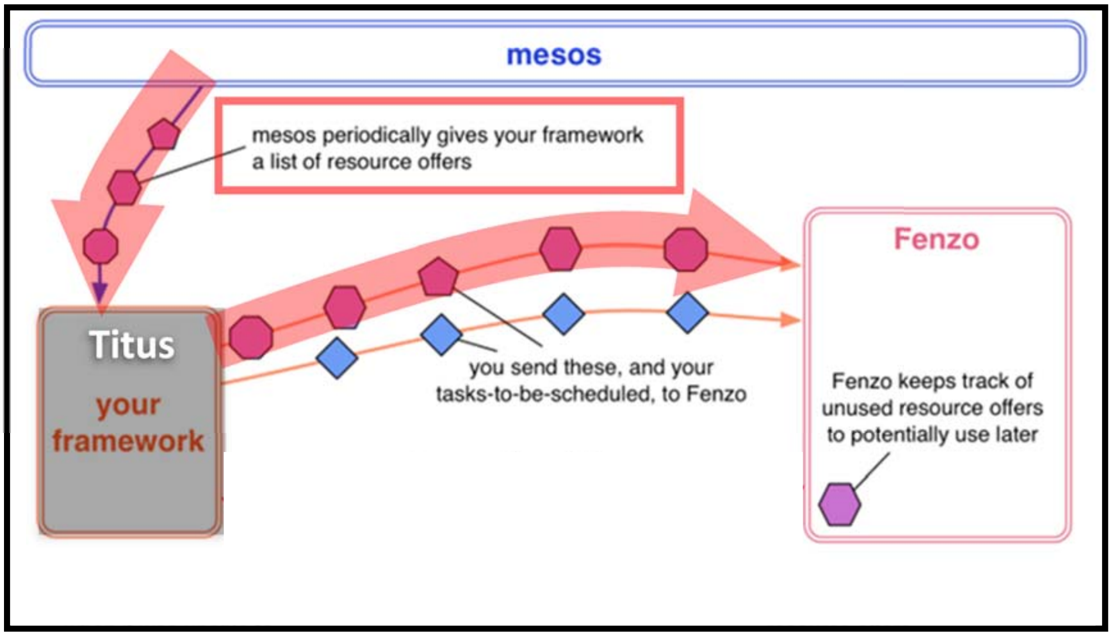
> Work in Titus is described by a job specification that details what to run (e.g., a container image and entry point), metadata (e.g., the job's purpose and who owns it), and what resources are required to run it, such as CPU, memory, or scheduling constraints (e.g., availability zone balancing or host affinity).[198]

382. Netflix also practices the step wherein the one or more processors are operable to "determine one or more utilization criteria for performing the job." For example, the Fenzo scheduler operating in conjunction with the Titus Master determines the utilization criteria for each task as well as for the available host resources. As Netflix explains:

> The Fenzo scheduler selects potential hosts [i.e., "Titus Agents"] to assign tasks to based on whether the hosts have sufficient resources to successfully execute the tasks. Each task has its own resource requirements, each host offer lists its available resources, and Fenzo maintains an accounting of which resources are available and which are already assigned on each host.[199]

383. Netflix also practices the step wherein the one or more processors are operable to "provide a list of available computer devices, the list comprising a plurality of computer devices currently provisioned to perform computer operations." For example, Netflix explains that "Fenzo takes as its input the resource offers that a framework [such as Titus] receives from Mesos and the tasks that are given to the framework."[200]

[198] *See* https://netflix.github.io/titus/overview/.
[199] *See* https://github.com/Netflix/Fenzo/wiki/Introduction.
[200] *See* https://github.com/Netflix/Fenzo/wiki.


mesos
mesos periodically gives your framework a list of resource offers
Titus
your framework
you send these, and your tasks-to-be-scheduled, to Fenzo
Fenzo
Fenzo keeps track of unused resource offers to potentially use later

Source: https://github.com/Netflix/Fenzo/wiki (annotations added).

384. Netflix also practices the step wherein the one or more processors are operable to, "from the list of available computer devices, determine a list of suitable computer devices for performing the job by comparing operational characteristics for each available computer device with the job criteria, the list of suitable computer devices comprising one or more computer devices having operational characteristics that satisfy the job criteria." For example, Mesos allows the Titus platform to set filters to reject resource offers that do not satisfy the desired operational characteristics in order to present a list of suitable resource offers.[201] For instance, a filter may include a constraint such that Mesos will "only offer nodes with at least *R* resources free" to the Titus Master.[202] Notably, after Mesos applies such filters to the resource offers, "the frameworks [i.e., the Titus/Fenzo

[201] Hindman, Benjamin, et al., *Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center*, NSDI, (2011) at p. 3-4 (available at https://www.usenix.org/legacy/events/nsdi11/tech/full_papers/Hindman.pdf).
[202] *Id.*

architecture] still ha[s] the ultimate control to reject any resources that [it] cannot express filters for and to choose which tasks to run on each node."[203]

As another example, the Fenzo scheduler itself applies multiple iterative constraints on the resource offers provided by Mesos via the Titus Master. Fenzo includes various built-in constraints that compare operational characteristics (such as available CPUs, disk space, memory, bandwidth, etc.) for each host to the job requirements in order to identify suitable resource offers.[204] As Netflix explains, a constraint evaluator inspects a target and applies any hard constraints (i.e., those that require "a simple Boolean thumbs-up or thumbs-down") to identify a list of suitable resource offers.[205] After applying these hard constraints, the Fenzo scheduler applies any soft constraints and uses any fitness calculators to further narrow the list of suitable resources.[206]

385. Netflix also practices the step wherein the one or more processors are operable to "use the utilization criteria to determine whether one or more underutilized computer devices exist on the list of suitable computer devices, the one or more underutilized computer devices having a suitable level of utilization for performing the job." As previously discussed, the Fenzo scheduler is configured to

[203] *Id*. at p. 4.

[204] *See* https://github.com/Netflix/Fenzo/wiki/Constraints; *see also* https://github.com/Netflix/Fenzo/wiki/Insights.

[205] *Id*.; *see also*, http://netflix.github.io/Fenzo/fenzo-core/com/netflix/fenzo/ConstraintEvaluator.html.

[206] *See* https://github.com/Netflix/Fenzo/wiki/Fitness-Calculators (explaining that the "VMTaskFitnessCalculator" interface provided by Fenzo "does not have to check to see that the proposed host has sufficient resources for the proposed task. It can assume that this has already been done.").

apply various hard and/or soft constraints to determine the "best fit" from the list of suitable resource offers.[207] As Netflix explains:

> The Fenzo scheduler selects potential hosts to assign tasks to based on whether the hosts have sufficient resources to successfully execute the tasks. . . . But frequently there are many hosts that meet these baseline qualifications. You can fine-tune how the Fenzo scheduler assigns tasks to compatible hosts by using fitness calculators and constraints.[208]

One such constraint is the "Balanced Host Attribute Constraint" that is built-in to Fenzo.[209] The Balanced Host Attribute Constraint "schedules tasks so that they are distributed evenly among types of hosts."[210] As Netflix explains, when applied as a soft constraint, the Balanced Host Attribute "weighs how close a host group would be to having the average number of tasks if a new task were assigned to it" in order to identify any underutilized resources.[211] Applying this constraint causes an underutilized resource to return a higher "fitness" rating for the new task.

386. Finally, Netflix practices the step wherein the one or more processors are operable to "if the one or more underutilized computer devices exist, identify the one or more underutilized computer devices to the client machine." For example, after Fenzo matches a task request with a resource offer, it provides its scheduling recommendations to the Titus Master.[212] As Netflix explains, the

[207] https://github.com/Netflix/Fenzo/wiki/Introduction.

[208] *Id.*

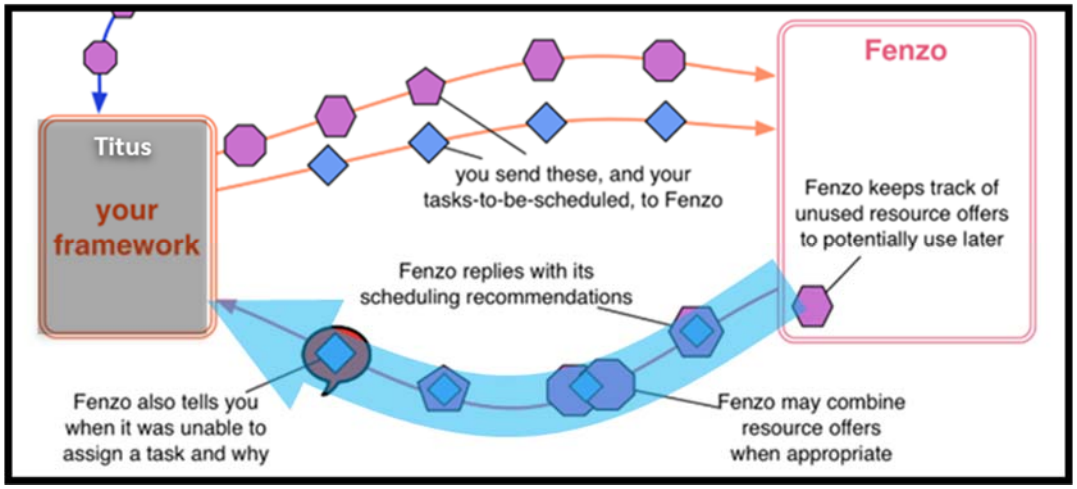
[209] *Id.*

[210] *Id.*

[211] *Id.*

[212] *See* https://netflix.github.io/titus/overview/.

Master then "schedules tasks onto Titus agents that launch containers based on the task's job specification."[213]


Titus
your framework
Fenzo
you send these, and your tasks-to-be-scheduled, to Fenzo
Fenzo keeps track of unused resource offers to potentially use later
Fenzo replies with its scheduling recommendations
Fenzo also tells you when it was unable to assign a task and why
Fenzo may combine resource offers when appropriate

Source: https://github.com/Netflix/Fenzo/wiki (annotations added).

387. With regard to claim 12 of the '183 Patent, Netflix practices "[t]he system of claim 11, wherein the one or more processors are operable to use the utilization criteria to identify one or more underutilized computer devices by, for each computer device on the list of suitable computer devices: determining a utilization value." For example, the Fenzo scheduler provides a built-in "Host Attribute Value Constraint" that "matches a task with a host that has a particular value [i.e., a utilization value] for a particular host attribute [such as CPU or disk utilization]."[214]

388. Netflix practices the system of claim 11, further comprising "comparing the utilization value with the utilization criteria." For example, as Netflix explains,

> When [the Host Attribute Value Constraint] evaluates a host to see if it is appropriate for a task, it calls that

[213] *Id.*

[214] https://github.com/Netflix/Fenzo/wiki/Constraints.

> function and then checks the value it returns against the value of the attribute on the host. If they match, the constraint is met; otherwise (or if the host does not have such an attribute) the constraint is not met.[215]

389. Finally, Netflix practices the system of claim 11, further comprising "removing the each device from the list of suitable computer devices if the utilization value does not satisfy the utilization criteria." For example, as Netflix explains:

> Fenzo will apply each constraint in this list to the hosts it evaluates for the task. If any of these constraints fail, Fenzo will disqualify that host from hosting the task and will move on to evaluating the next host.[216]

390. With regard to claim 15 of the '183 Patent, Netflix practices "[t]he method of claim 11, wherein the one or more processors are configured to: use the utilization criteria to determine whether one or more underutilized computer devices exist on the list of suitable computer devices by identifying a computer device having a lowest level of activity relative to other computer devices on the list of suitable computer devices." For example, as previously discussed, the Balanced Host Attribute Constraint used by Fenzo "schedules a set of tasks so that they are distributed evenly among types of hosts."[217] As Netflix explains, "this constraint attempts to minimize the difference between the number of co-tasks running on the host type with the *most* co-tasks running on it and the number of co-tasks running on the host type with the *least* co-tasks on it."[218] This constraint may

[215] *Id.*
[216] *Id.*
[217] *Id.*
[218] *Id.*

be used to identify the host having the lowest level of activity relative to other suitable computer devices. As Netflix explains, the Balanced Host Attribute Constraint identifies underutilized hosts by assigning each host a fitness score between 0.0 (not a fit) to 1.0 (perfect fit).[219]

391. Finally, Netflix practices the system of claim 11 wherein the one or more processors are configured to "identify the computer device having the lowest level of activity to the client machine." For example, as discussed above, the Best Host Attribute Constraint will assign the highest fitness score to the resource with the fewest tasks. After the Fenzo scheduler matches a task request with a resource offer that complies with any applicable constraints (including, for example, the "Balanced Host Attribute Constraint"), it provides a recommendation to the Titus Master.[220] As Netflix explains, the Master then "schedules tasks onto Titus agents that launch containers based on the task's job specification."[221]

392. Upon information and belief, Netflix directly infringes other claims of the '183 Patent as well, including for the reasons discussed in the preceding paragraphs.

393. At least as of the filing and service of this Amended Complaint, Netflix has had knowledge of the '183 Patent and Netflix's infringement thereof.

394. Netflix's infringement of the '183 Patent, which is knowing and willful at least as of the filing of this Amended Complaint, has caused and continues to cause damage to Avago. Avago is entitled to recover damages sustained as a result of Netflix's wrongful acts in an amount subject to proof at trial.

[219] *Id.*

[220] *See* https://netflix.github.io/titus/overview/.

[221] *Id.*

PRAYER FOR RELIEF

WHEREFORE, the Broadcom Entities respectfully request that the Court enter a judgment in their favor and against Netflix:

1. Declaring that Netflix has directly infringed one or more claims of the Patents-in-Suit in violation of 35 U.S.C. § 271;

2. Declaring that Netflix has induced infringement of one or more claims of the ’079, ’121, ’245, and ’992 Patents in violation of 35 U.S.C. § 271(b);

3. Declaring that Netflix’s infringement of the ’079, ’121, ’245, ’992, ’138, ’387, ’663, ’283, ’976, ’722, and ’183 Patents is willful and deliberate pursuant to 35 U.S.C. § 284;

4. Enjoining Netflix from further infringing the ’079, ’121, ’245, ’992, ’138, ’976, ’722, and ’183 Patents;

5. Ordering that the Broadcom Entities be awarded damages in an amount no less than a reasonable royalty for each asserted patent arising out of Netflix’s infringement of the Patents-in-Suit, together with any other monetary amounts recoverable, such as treble damages;

6. Declaring that this is an exceptional case under 35 U.S.C. § 285 and awarding the Broadcom Entities their attorneys’ fees and costs;

7. Ordering that Netflix is required to pay exemplary damages pursuant to 35 U.S.C. § 284;

8. Awarding pre-judgment and post-judgment interest and costs against Netflix; and

9. Awarding the Broadcom Entities such other and further relief as the Court deems just and proper.

JURY DEMAND

The Broadcom Entities demand a trial by jury of all claims in this action so triable.

Dated: June 22, 2020

HOPKINS & CARLEY, A Law Corp.

By: */s/ Christopher A. Hohn*
Christopher A. Hohn
Attorneys for Plaintiffs
BROADCOM CORPORATION and
AVAGO TECHNOLOGIES
INTERNATIONAL SALES PTE.
LIMITED