BitTorrent

- توی این اپلیکیشن ، یه فایلی برای توزیع شدن به قطعات(chunks) 256Kb تبدیل میشه و peer ها داخل یک torrent این file chunk ها رو با هم رد و بدل می کنن.
- Torrent به گروهی از peer ها که chunk های مختلف یک فایل رو باهم رد و بدل می کنن اطلاق میشه.
- BitTorrent یه پروتکل pure P2P نیست، یعنی نیاز به یه minimum infrastructure ای داره که بهش tracker گفته میشه.
- Tracker یه سروریه که وقتی یه peer جدیدی می خواد یه فایلی رو از طریق BitTorrent دریافت کنه ، به این سرور مراجعه می کنه، خودش رو معرفی می کنه (register) و یه لیستی از peer هایی که توی این torrent برای دریافت این فایل وجود دارن، دریافت می کنه.این لیست می تونه به طور تصادفی هم انتخاب بشه، چون تعداد peer ها ممکنه خیلی زیاد باشه.

کار دیگه ای که tracker انجام میده اینه که مشخصات اون chunk ها رو هم در اختیار peer جدید قرار میده که مثلا این فایل از چندتا chunk تشکیل شده و هر chunk مشخصاتش چی هست.

به این ترتیب peer جدید، میدونه که برای دریافت کل فایل باید چه chunk هایی رو دریافت کنه.

- یک peer ای که در ابتدای کار به یه torrent ملحق میشه هیچ chunk ای نداره، به تدریج تعداد chunk هایی که دریافت می کنه بیشتر و بیشتر میشه.

بعد از اینکه لیست peer های این torrent (neighbors) رو دریافت کرد،به صورت همزمان با هر کدوم از اون ها یه ارتباط TCP تشکیل میده.

- همزمان با دانلود کردن، باید آپلود chunk هم توسط یه peer صورت بگیره.اگه این سرویس فعال نباشه، peer ها ممکنه به صورت selfish عمل کنن یعنی فقط تقاضای دریافت فایل رو بدن ولی هیچ فایلی آپلود نکنن و در اختیار peer های دیگه قرار ندن.
- Peer ها وضعیت ثابتی ندارن و ارتباطشون با شبکه ممکنه قطع و وصل بشه، به همین دلیل، همسایه های یک peer میتونن تغییر کنن، و لیست همسایه های یک peer ، فیکس نیست.

- Requesting chunks :

- سوالی که پیش میاد اینه که وقتی یه peer به torrent متصل شد، اول برای کدوم chunk ها درخواست بده؟

- کاری که توی پروتکل BitTorrent انجام میشه اینه که به صورت periodic ما از همسایه هامون درخواست می کنیم که لیست chunk هایی که دارن رو برای ما ارسال کنن، بعد نگاه می کنیم ببینیم اون chunk هایی که ما میخوایم رو کیا در اختیار دارن، و در ضمن بین chunk هایی که نداریم،کدوم کمیاب تره و اون هارو درخواست میدیم.
به این روش rarest first گفته میشه.
این استراتژی منطقی هست چون باعث میشه chunk هایی که کمیاب هستن تعدادشون در torrent بیشتر بشه.

- Sending chunks :

- استراتژی ای که برای ارسال chunk ها استفاده می کنیم tit-for-tat نام داره .
یعنی ما میایم همسایه هامون رو براساس سرویسی که اخیرا به ما دادن، و با سرعت بالاتری برای ما chunk فرستادن، رنک می کنیم. اون ۴ تایی که در بالای لیست قرار می گیرن انتخاب میشن تا ما برای اون ها chunk ارسال کنیم. این لیست هر ۱۰ ثانیه آپدیت میشه.
این استراتژی باعث میشه peer ها انگیزه ی لازم برای مشارکت رو پیدا کنن ، به خاطر این که اگه یه peer ای چیزی آپلود نکنه یا با سرعت کمی آپلود کنه ، باعث میشه توی short list سایر peer ها قرار نگیره و نتونه از سرعت بالای down link بهره مند بشه.

- این استراتژی یه تبصره داره اونم اینه که هر ۳۰ ثانیه یه peer ای به صورت تصادفی از بین همسایه ها انتخاب میشه و اون هم جزو peer هایی قرار می گیره که ما براش chunk ارسال می کنه . به این کار، Optimistically unchoke کردن گفته میشه.

(choke کردن یعنی مسدود کردن و unchoke کردن یعنی باز کردن)

دلیل این کار هم اینه که اولا به یه سری همسایه ها که همدیگه رو نمیشناسن فرصت داده بشه ، دوما در ابتدای کار که هیچ chunk ای در اختیار نداریم، بتونیم به صورت تصادفی توی لیست سرویس peer های همسایه قرار بگیریم تا بتونیم chunk جمع آوری کنیم و وقتی توی آپلود کردن chunk مشارکت می کنیم،بتونیم توی ۴ تای اول لیست قرار بگیریم.

- پسوند فایل هایی که توسط BitTorrent دانلود می کنیم، .torrent هست و بعد از دانلود ، باید این فایل رو به یه کلاینتی که می تونه اپلیکیشن BitTorrent رو اجرا کنه ، به عنوان ورودی بدیم تا دریافت فایل به صورت توزیع شده توسط این اپلیکیشن انجام بشه.

Video Streaming and CDNs

- Video streaming یه سرویس خیلی مهمه و پیش بینی شده بود که تا سال ۲۰۲۰ ، ۸۰٪ ترافیک اینترنت مربوط به شبکه هایی بشه که این سرویس رو ارائه میدن، مثل Netflix ، YouTube ، Amazon Prime و ...
- چالشی که این کمپانی ها باهاش دست و پنجه نرم می کنن ، مقیاس هست ، چون تعداد کاربرانشون خیلی زیاد و در حد ۱ میلیارد نفره.
همچنین یه چالش دیگه اینه که چون کاربران از جاهای مختلف هستن و شرایط متفاوتی دارن، access network هاشون هم متفاوته و پهنای باند یکسانی ندارن. پس این شرکت ها باید تمهیداتی انجام بدن که هر کاربری با هر شرایطی بتونه از سرویس هاشون بهره مند بشه.
روش اصلی ای که این شرکت ها برای مقابله با چالش ها به کار می گیرن، اینه که از یه infrastructure ِ توزیع شده در سطح application استفاده می کنن. یعنی اون سرور هایی که قرار هست ویدیوها در اون ها ذخیره بشه و به تقاضای کاربرا پاسخ بدن، همگی توی یک دیتاسنتر نیستن ، بلکه به صورت توزیع شده در جاهای مختلف قرار می گیرن.به این ترتیب این دوتا چالش تا حدودی رفع میشه.

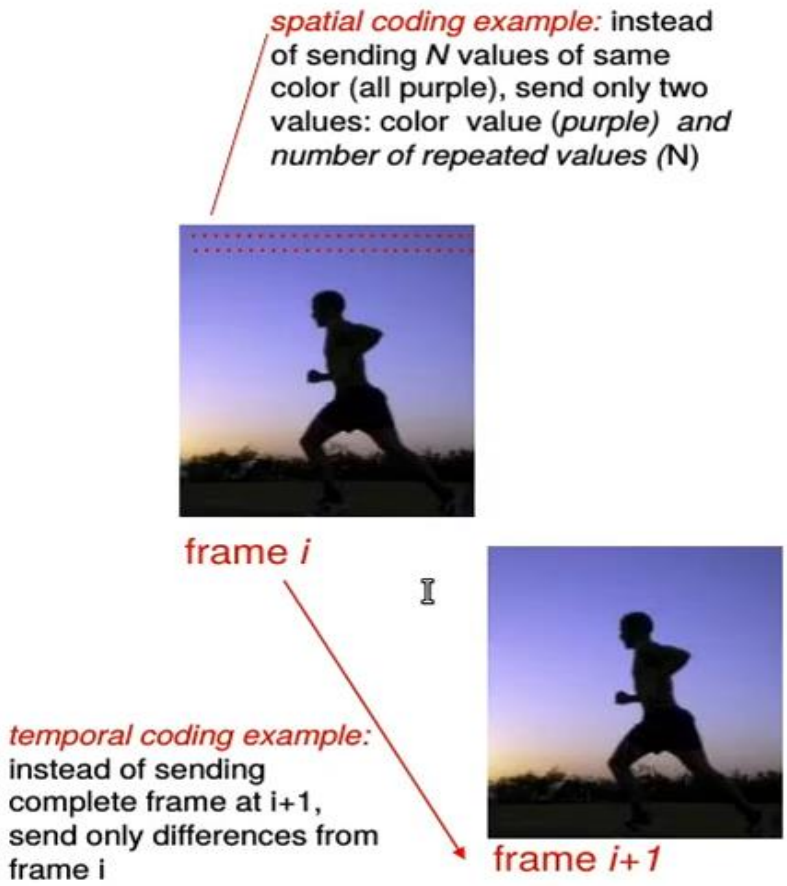
- Video

- Video یک رشته از تصاویره. وقتی یک رشته از تصاویر با یک بیت ریت ثابت (مثلا 24 image/sec) نمایش داده میشه ، یک video شکل می گیره.
- وقتی میخوایم یه image رو به صورت دیجیتالی نمایشش بدیم، به یک ماتریس به تعداد پیکسل هایی که به صورت سطری و ستونی اون عکس رو تشکیل میدن نیاز داریم، و بعد اطلاعات هر پیکسل (مثلا اون پیکسل چه رنگی داره) رو با یک سری بیت نشون میدیم.
پس جنس داده ی image مثل یک ماتریسی از اعداده که هر کدوم از اون اعداد رو به یه تعداد بیت نمایش میدیم.
- برای اینکه بتونیم تو پهنای باند صرفه جویی کنیم و به صورت Adaptive عمل کنیم، میایم رشته ی تصاویر رو کد می کنیم.یعنی سعی می کنیم تا اونجایی که میشه حداقل تعداد بیت رو استفاده و ارسال کنیم. در واقع افزونگی ها(redundancy) هایی که در داخل یه عکس یا بین عکس ها وجود داره رو تا اونجا که ممکنه کاهش میدیم.
- برای فشرده سازی، از دو روش می تونیم استفاده کنیم:

1 - استفاده از هم بستگی و شباهت در حوزه ی مکان در داخل یک تصویر . یعنی پیکسل های مجاور یک تصویر باهم شباهت هایی دارن

که ما میتونیم ازش استفاده کنیم تا تعداد بیت های کمتری برای انکود کردن یه تصویر به کار ببریم.(spatial)

2 - تو حوزه ی زمان هم می تونیم از شباهت یک تصویر با تصویر بعدی استفاده کنیم.(temporal) پس به جای اینکه تک تک تصاویر رو انکود کنیم و بفرستیم، تفاوت های اون ها رو انکود می کنیم و ارسال می کنیم که باعث صرفه جویی در پهنای باند و بیت ریت میشه.



- بعضی از روش های coding برای video ، از ویژگی های temporal coding و spatial coding استفاده نمی کنن و به همین دلیل بیت

ریتشون فیکسه و تغییری نمی کنه.بهشون CBR(constant bit rate) گفته میشه.

اما بعضی از روش ها هم هستن که ازین دو روش استفاده می کنن و بیت ریتشون تغییر می کنه که VBR(variable bit rate) نام دارن.

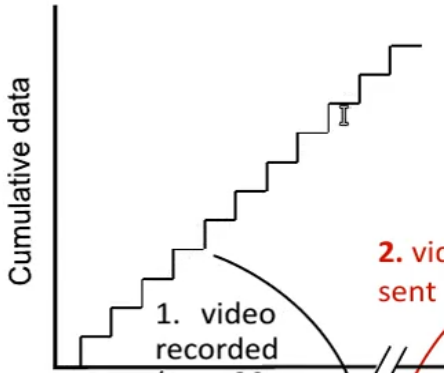
توی روش VBR ، قدرت فشرده سازی خیلی بستگی داره به اینکه جنس تصاویر چیه ، شباهت پیکسل های یک تصویر چقدره، میزان شباهت فریم های متوالی به چه شکله و ... .

مثال از انواع video coding ها:

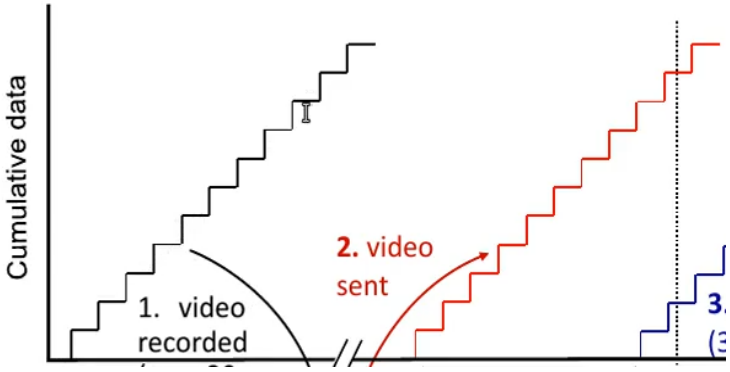
- MPEG 1 (CD-ROM) 1.5 Mbps
- MPEG2 (DVD) 3-6 Mbps
- MPEG4 (often used in Internet, 64Kbps – 12 Mbps)

- مهم ترین چالش اینترنت برای سرویس video streaming ، اینه که پهنای باند لینک بین سرور و کلاینت ، با گذشت زمان تغییر می کنه و ترافیک شبکه هم بسته به شرایط تغییر می کنه . این عوامل میتونه باعث packet loss و تاخیر های متغیر با زمان بشه.

- مثال : نرخ فریم برداری در یه ویدیو 30 frame/sec هست.به این ترتیب نمودار حجم داده بر حسب زمان به صورت پلکانی میشه(نمودار مشکی) :

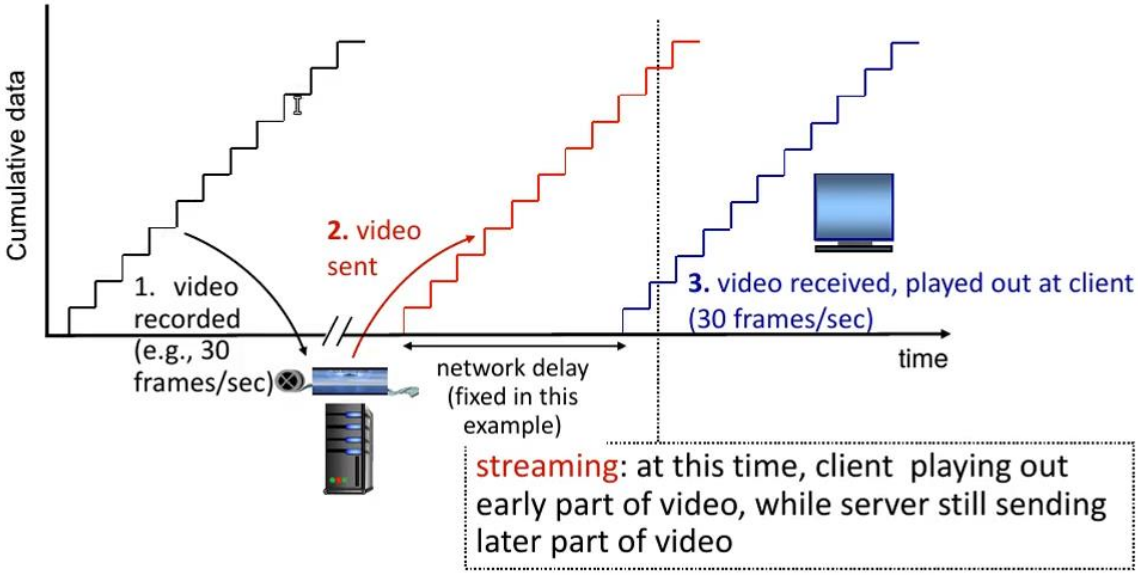


وقتی یه کلاینت از سرور تقاضای دریافت این ویدیو رو می کنه ، مجددا نمودار حجم داده بر حسب زمان به شکل پلکانی میشه و ویدیو بر حسب timestamp هر فریم ارسال میشه (نمودار قرمز) :



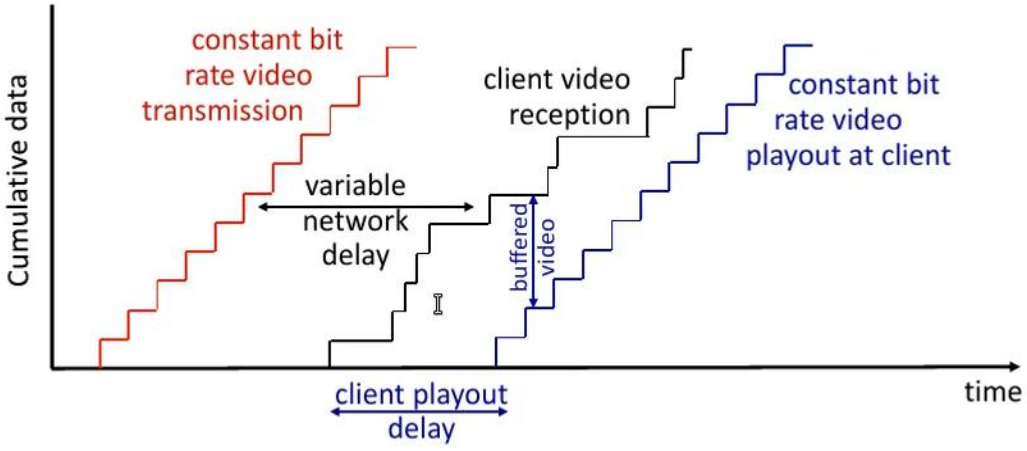
اگه شبکه ایده آل باشه و تاخیرش فیکس باشه ، با همون شکلی که سرور داده رو ارسال کرده کلاینت هم به همون شکل داده رو دریافت می کنه

و فقط یه شیفت زمانی رخ داده . بنابراین سمت کلاینت بدون اینکه زیاد منتظر بمونیم فریم ها رو یکی پس از دیگری دریافت می کنیم(شکل آبی):



اما در عمل این طوره که تاخیر لینک بین سرور و کلاینت و پهنای باند متفاوته. به تاخیر متغیر با زمان jitter هم میگن.

توی این شرایط کاری که انجام میشه برای تاخیر متغیر، استفاده از buffer هست :



اگه تاخیر بین فریم ها در زمان ارسال متغیر باشه (مشابه شکل آبی) دیگه اون ترتیب زمانی فیکس در اون لحاظ نشده. اما میتونیم فریم هایی که از قبل دریافت شدن رو در بافر ذخیره کنیم و اجازه بدیم اول چند تا فریم دریافت بشن ، بعد ویدیو رو پخش کنیم.

فاصله ی عمودی بین نمودار مشکی(حجم ارسالی ویدیو بر حسب زمان سمت سرور) و نمودار آبی(حجم ویدیوی پخش شده برحسب زمان سمت کلاینت) در هر زمان، حجم فریم های ذخیره شده در بافر رو نشون میده.

- مشکل کمبود و تغییر پهنای باند رو دیگه نمیشه با بافر سمت کلاینت حل کرد. این مشکل نیازمند پروتکل های خاص و شبکه های Delivery network هست.

  پروتکلی که در این راستا وجود داره ، DASH(Dynamic, Adaptive Streaming over HTTP) هست.

  توی این پروتکل ، ویدیو، در سمت **سرور**:

  1 - به تعدادی chunk تقسیم می شه،

  2 - هر chunk با ریت های مختلف(متناسب با پهنای باند های مختلف) کد گذاری میشه،

  3 - کپی این فایل ها در CDN node های مختلف قرار می گیرن و فقط یه نسخه از این فایل نداریم.

4 - به جز chunk ها ، فایلی تحت عنوان manifest داریم که URL مربوط به هر chunk ، اطلاعات مربوط به code rate هایی که از هر chunk موجوده و اینکه روی کدوم یک از سرور های CDN میتونیم پیداش کنیم رو در داخل خودش داره .

سمت کلاینت :

1 - پهنای باند server-to-client به طور متناوب میزانش سنجیده میشه.

2 - با استفاده از اطلاعات فایل manifest ، chunk هایی که اون ویدیو هست رو در خواست میده.

هر بار متناسب با پهنای باندی که داریم ،تقاضای دریافت coding rate متناسب با اون رو میدیم.

در حالت کلی ،هر chunk رو میتونیم ورژن های مختلفش رو براساس code rate های مختلف درخواست بدیم، یا اینکه از سرور های مختلف تقاضا کنیم.

با استفاده از این کار ها میشه مشکل پهنای باند رو تا حدود خوبی حل کرد.

- توی روش DASH کلاینت این هوشمندی رو داره که :

1 - چه موقع باید chunk درخواست بده . نه اون قدر تند که بافر سمت کلاینت overflow کنه و نه اون قدر تاخیر داشته باشه که بافر خالی بشه و پخش فریم ها دچار مشکل بشه.

2 - چه encoding rate ای رو با توجه به براوردی که از پهنای باند سرور های CDN داره، درخواست بده.

3 - از چه سروری درخواست chunk کنه. ممکنه سرور نزدیک تر رو انتخاب کنه چون تاخیر کمتری داره، یا سروری که دورتره ولی ترافیک شبکه ش کمتره .(در هر صورت چیزی که تعیین کننده است، پهنای بانده)

پس به طور کلی تکنیک هایی که در سرویس video streaming استفاده میشن اینها هستن:

Streaming video = encoding + DASH + playout buffering

- Content distribution networks(CDNs)

- خیلی نقش مهمی در سرویس video streaming ایفا می کنن.
- چالش : چه طور محتوایی مثل یک ویدیو رو در اختیار تعداد زیادی کاربر(در حد چندصد هزار) قرار بدیم؟

راه حل اول : به صورت متمرکز عمل کنیم. یعنی داخل یک دیتاسنتر، چند تا سرور وجود داشته باشه و ازین سرور ها برای ارسال ویدیو استفاده کنیم. این راه چندتا مشکل داره :

1 - اگه دیتاسنتر با مشکل مواجه شد اون سرویس هم با مشکل مواجه میشه (single point of failure)

2 - ترفیک بالای شبکه (point of network congestion)

3 - تاخیر به خاطر این که دیتاسنتر از جاهای مختلف فواصل متفاوتی داره و ممکنه دور باشه.

راه حل دوم : کپی های زیادی از فایل رو در سرور های مختلف در نقاط مختلف دنیا قرار میدیم.(Content distribution networks(CDNs))

- در مورد اینکه سرور ها رو در چه نقاطی قرار بدیم دوتا استراتژی وجود داره :

1 - Enter deep : سرور ها رو در نزدیک ترین نقطه به کاربران،یعنی در access network ها قرار بدن. شبکه های CDN شرکت Akamai ازین استراتژی استفاده می کنن و حدود 240000 سرور در بیشتر از 120 کشور دنیا قرار دادن.

2 - Bring home : توی این استراتژی میزان پخش شدگی کمتر هست، و cluster هایی از سرور های مربوط به CDN ها، در POP

(points of present) های ISP ها قرار داده می شن. یه مقدار فاصله ی سرور ها از کاربران بیشتر میشه ولی مدیریت CDN ها سبک تر میشه به خاطر اینکه میزان توزیع cluster های سرور ها کمتره.

شرکت Limelight ازین استراتژی استفاده می کنه.

- یکی دیگه از شرکت هایی که سرویس CDN ارائه میده ، Cloudflare هست. توی وب سایتش(cloudflare.com) نشون میده که سرور هاش از لحاظ جغرافیایی کدوم مکان ها رو پوشش میدن. در حال حاضر سرویس های این شرکت بیشتر از 200 شهر در بیشتر از 100 کشور دنیا رو پوشش میده.
- یک مثال از شرکت هایی که در حوزه video streaming فعالیت می کنه ، Netflix عه.این شرکت یه CDN داره. اگه یک فیلم از مجموعه ی فیلم هاش رو در نظر بگیریم(مثلا MADMEN) نسخه هایی ازین فیلم در سرور های مختلف و در جاهای مختلف وجود داره. حالا اگه کاربری بخواد این فیلم رو مشاهده کنه ، درگام اول به وب سایت Netflix مراجعه می کنه و درخواست میده که فایل manifest رو براش ارسال کنه(به عبارتی می پرسه که این فیلم رو از کدوم سرور ها و با چه بیت ریتی میتونم دریافت کنم).

بعد ، وب سرویس Netflix ، فایل manifest رو برای کلاینت ارسال می کنه و مبتنی بر این فایل ، کاربر میتونه سروری که بهترین لینک رو

باهاش داره پیدا کنه و درخواست دریافت chunk می کنه ازش.(با بیت ریتی متناسب با لینک بینشون)
اگه کیفیت لینک به هر دلیلی پایین اومد، (مثلا ترافیک شبکه بالا رفت) کلاینت هوشمندی لازم رو داره که از بین سرور های دیگه ، سروری که بهترین لینک رو داره انتخاب کنه و chunk ها رو از اون سرور دریافت کنه.

- به CDN ها ، OTT(over the top) هم اطلاق میشه.

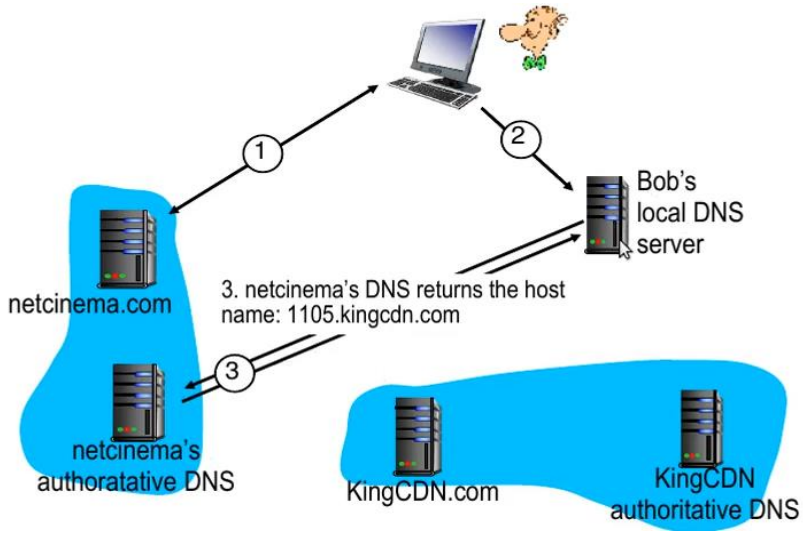
به دلیل اینکه در CDN ، به کل اینترنت به عنوان یه سرویس ارتباطی نگاه میشه که CDN قصد داره بر روی این بستر ارتباطی ، یه سرویس دیگه ارائه کنه. پس CDN ها باید چالش هایی که در اینترنت وجود داره رو به نحوی توسط سرور هایی که در edge قرار دارن، رفع کنن.
یکی ازین چالش ها ، اینه که چه محتوایی رو روی کدوم CDN node قرار بدن؟(چون نمیشه همه ی محتواها رو روی همه ی cluster ها داشته باشیم)
یکی دیگه ازین چالش ها، اینه که چه طور باید بفهیم که یه محتوا، روی کدوم node قرار گرفته و اطلاعات رو با چه ریتی و از چه سروری باید بگیریم؟
یه روش پاسخ به این سوال ها، توی مثال قبل بیان شد که استفاده از فایل manifest و هوشمندی کلاینت ها بود.

یه روش دیگه برای پاسخ به این سوال ها ، روش مبتنی بر استفاده از DNS هست.

مثال : فرض می کنیم یه شرکت هست به اسم net cinema که سرویس video streaming ارائه میده. این شرکت خودش CDN نداره و از یه CDN provider (به نام KingCDN) استفاده می کنه تا بتونه کپی های مختلف از فایل هاش رو به جاهای مختلف بفرسته. یک مشترک به وب سایت این شرکت مراجعه می کنه و درخواست دریافت یه ویدیو رو می کنه و این URL بهش برگردونده میشه :

http://video.netcinema.com/6Y7B23V

چون این URL ، hostname اش با video شروع میشه (با netcinema.com متفاوته) ، browser مجبور میشه که از Local DNS server استفاده کنه تا آدرس IP مربوط به این URL رو پیدا کنه. این Local DNS server هم به نیابت از کلاینت میره از authoritative DNS server سوال می پرسه.ولی این سرور ، به جای آدرس IP ، میاد یه hostname مربوط به شرکت KingCDN رو برمی گردونه :



بنابراین Local DNS server میره از authoritative DNS server شرکت KingCDN سوالش رو می پرسه. این سرور هم باید تصمیم گیری کنه که ارتباط کلاینت رو با یکی از سرور هایی که محتوای شرکت net cinema توش هست ، برقرار کنه.معیار های مختلفی مثل توازن بار یا نزدیکی سرور به کلاینت ، در این تصمیم گیری تاثیر دارن. نهایتا IP یکی از این سرور ها رو برای کلاینت می فرسته و کلاینت ویدیوی خودش رو ازین طریق دریافت می کنه.