Tcp خودش به صورت اولیه sockets نبودند .

در Http چگونه ارتباط بین client و server برقرار می شود :

هر صفحهٔ وبسایتی که شما دارید از چندین object تشکیل شده که این object ها اپلیکیشن HTML_file پایه است . که پایه وبسایت روی سازه . مثلاً چندین عکس یا شکل یا ویدیو باشه . که هر اینها object ها روی سازند .

دو نوع روش HTTp داریم → Non-persistent HTTp

persistent HTTp

Non-persistent : که به صورت object به object می خواد ارسال کنه برای هر Tcp یه connection کانکشن بزنه

① بازکردن Tcp connection
② فرستادن یک object
③ بستن Tcp connection

persistent : یه Tcp کانکشن ابتدایی می زنه و همه objectها رو وجود داره رو درخواست داره رو یک به یک بدون Tcp conne مجدد انجام می دهد.

که زبان صحبت کردن با server رو تعریف می کنه.

① بازکردن Tcp conn
② فرستادن چندین object
③ بستن Tcp conne

به persistent ، HTTp 1.1 گفته می شود.

HTTP یک پروتکل stateles هست و صحبت های که بین client و server

انجام شده در هیچ جای ذخیره نمی شود که اگر این کار را بکنند خیلی پیچیده می شود

و owerhade (سربار) خیلی بالای دارد به همین دلیل این کار انجام نمی شه.

یک راهی که می تونه انجام بشه مسئله cookies هست: اینطوره که اون server که هست

و server HTTP هم هست میاد و از client می خواد که ~~اگر~~ اگه تمایل داره بپذیره که

یک سری از اطلاعاتش ذخیره بشه. اگرچه به طور عادی یک سری از اطلاعات و cookies های که اجباری هستند

بدون نیاز به هماهنگی و توافق باید ذخیره شوند. اما یه سری کارهای که مثل تبلیغات هست

رو باید هماهنگی برقرار بشه که ذخیره بشه یا نه. که کاربردهای مختلفی داره که مهمترینش بحث

تبلیغات هست که طرف رو می شناسه و به اون داده های مختلفی رو پیشنهاد میده.

موارد استفاده از cookies

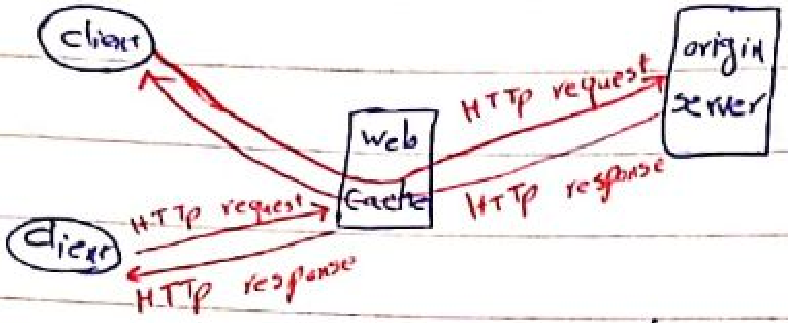
① authorization

② shopping carts

③ recommendations

④ user session state (web e-mail)

راه دیگه ای رو وجود داره و ما می تونیم وضعیت HTTP رو بهتر کنیم ~~web~~ web caches



به مشتری

web caches این طوری هست، به جای اینکه مستقیماً client با server ارتباط برقرار کنه

معمولاً در یک موسسه accsese point رخ میده. و ما می تونیم web cache رو روی

browser فعال کنیم. و کاری که می کنه اینه که: نگاه می کنه اگر object در web cach

باشه که به client می فرسته، و اگر نباشه از server می گیره و می فرسته.

ولی باید تمام کانتنت (content)ها رو ذخیره کنه.

web cache هم به صورت client و هم به شکل server عمل می کنه.

وقتی web cache می خواد یه چیزی رو از server اصلی cache کنه server اصلی

بهش زمان میده که چند ثانیه باشه: cache-control: max-age=<seconds>

یا میگه این قابل condition cache کردن نیست: cache-control: no-cache

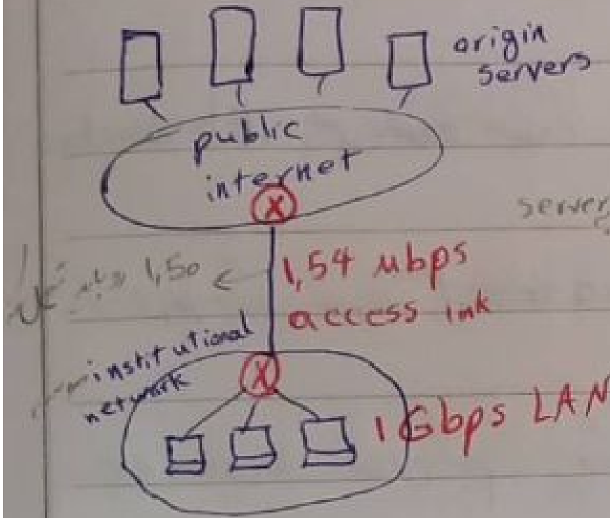
چرا از web cache ?

① response time رو کاهش میده چون نزدیکتره

② ترافیک را کاهش می دهد.

③ قابل استفاده برای content های ضعیف هم هست یا کسانی که نمی خواهند هزینه کنند

مثال:



accesslink rate = 1,54 mbps

RTT = 2 sec ← رفت و برگشت از server, clien

web object size = ۱۰۰ kbit

average request rate = 15 sec

average data rate = 1,50 mbps

۱۰۰×۱۵ = 1,50 mbps ← به این میزان پهنای باند داریم.

* چون 1,54 و 1,50 خیلی به هم نزدیکند امکان درخواست مجدد وجود ندارد.

* $\frac{1,50}{1,54} = ۰,۹۷$ ⇐ ترافیک ایجاد میشه و

* access link utilization = ۰,۹۷ → خوب نیست

و delay خیلی بالا میره.

* LAN utilization = ۰,۰۰۱۵ → $1,50 \times ۱۰^{-۳}$ → خیلی پایینه

* end-end delay = $\overbrace{۲\ sec}^{RTT} + \overbrace{۰,۹۷}^{minutes} + \overbrace{۹,۰۰۱۵}^{usecs}$

— می توان یه کار ساده لوحانه رو انجام داد می توان پهنای لینک (mbps)1,54 رو افزایش بدیم و هزینه زیاده که utilization پایین میاد و تأخیر ms میشه، و در نتیجه تأخیر کل به ثانیه (۲ sec) می رسه.

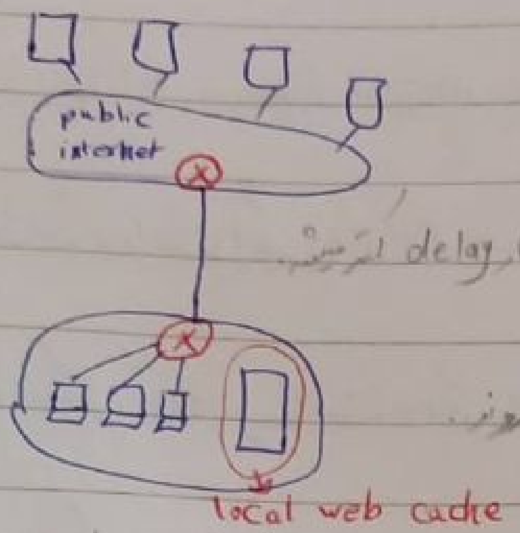
(اما خیلی گرونه)

* اما راه منطقی اینه از web cache استفاده کنیم که اطلاعات رو نگه میداره

storeg اضافه میکنه ruter و

هزینه اولیه ارتقاعش میره بالا اول ولی این کار delay کمتر میشه.

اما یه بخشی از requestها سراغ web cach میرن.



که اینجا یه مفهومی وجود داره به نام Hit rate : مشخص میکنه که در request

که بین ⊗ چقدر web cach رو hit میکنه؟

* hit rate = ۰٫۴ ← یعنی ۴۰٪ درخواست‌ها که میان ⊗ داده موردنظرشون در □ هست. و ۶۰٪ (۰٫۶) درخواست‌ها میرن بالا.

* rate to browser = ۰٫۶ x 1,50 = ۰٫۹ Mbps

* access link utilization = ۰٫۹ /1,54 = ۰٫58

یعنی utilization پایین تر میاد. وضعیت بهتری پیش میاد و delay کمتر میشه.

* average end-end delay : ۰٫۶ (۲٫۰۱) + ۰٫۴ (~ msec) ≅ ۱٫۲ sec

conditional GET : که بهش یه جاهایی cache هم گفته میشه.

وقتی client میاد HTTP request رو میفرسته میگه اون object رو که الان میخوام قبلاً داشتمش. اگر تا یه تاریخی عوض نشده همونو استفاده کنم. :)

و اگر ok باشه کد ۳۰۴ رو ارسال میکنه و client همون قبلی رو استفاده میکنه.

Year. Month. Date. ( ) Subject :

اگه اینطوری نباشه که دیگری رواش ارسال میکنه.

HTTP 1.1 : چندتا object رو می‌فرستد و مجموعاً pipelined GET ایجاد می‌کرد.

first come first served

نکته‌ای که داشت اینه FCFS هست و object ها رو FCFS می‌فرستاد.

که مشکل آن این است که ممکن است یه object بزرگ جلوی یه object کوچیک رو

بگیره. که به این اتفاق (head-of-line (HoL) blocking)

گفته می‌شود.

HTTP/2 : اما در HTTP2 سرور میاد و اول object بزرگه رو که اول صف

هست میشکنه و یه مقدار حداکثری برای هر object مشخص می‌کنه.

به صورت rund rubin اجرا می‌شه. که وضعیت عملکردی خیلی بهتر میشه.

E-mail : سه قسمت داره
① user agent ← خودما هستیم
② mail server ← gmail و ...
③ پروتکل SMTP : ① هنگام ارتباط ما با mail server
② هنگام ارسال ایمیل از mail server به گیرنده gmail داره.

کاربرد داره

TALASH

DNS : یه وقتی ما می خواهیم یه آدرس وبسایت رو بخواهیم مشاهده کنیم که این آدرس به آن url مشخص میشه. اما وقتی url یه چیزی رو میزنیم، می دونیم چه طوری باید بهش وصل بشیم؟ چون که هاست هایی که وجود دارند براساس Ip مشخص شده اند و وقتی که url یه چیزی رو می زنیم باید بدونیم به کدوم server باید rout بشیم به همین دلیل اول کسی هست که این کار را برای ما انجام میده DNS و mapping بین Ip و username رو شکل میده. و DNS ها قاعدتاً یه سری Database هستند. که مشخص می کنه یه url برای کدام Ip هست. و در لایه application-layer کار می کنه.

خدمات DNS:

① تغییر hostname به Ip آدرس

② host aliasing ← یه زمانی هست که یه host name واقعی نباشه یا host name دقیق تری وجود داشته باشه. که این ترجمان هم توسط DNS انجام می

③ mail server aliasing

④ load distribution ← ممکنه یه دونه url چندین آدرس Ip داشته باشه. که بین این ها تقسیم میکنه که load کمتر بشه. از وظایف DNS هست.

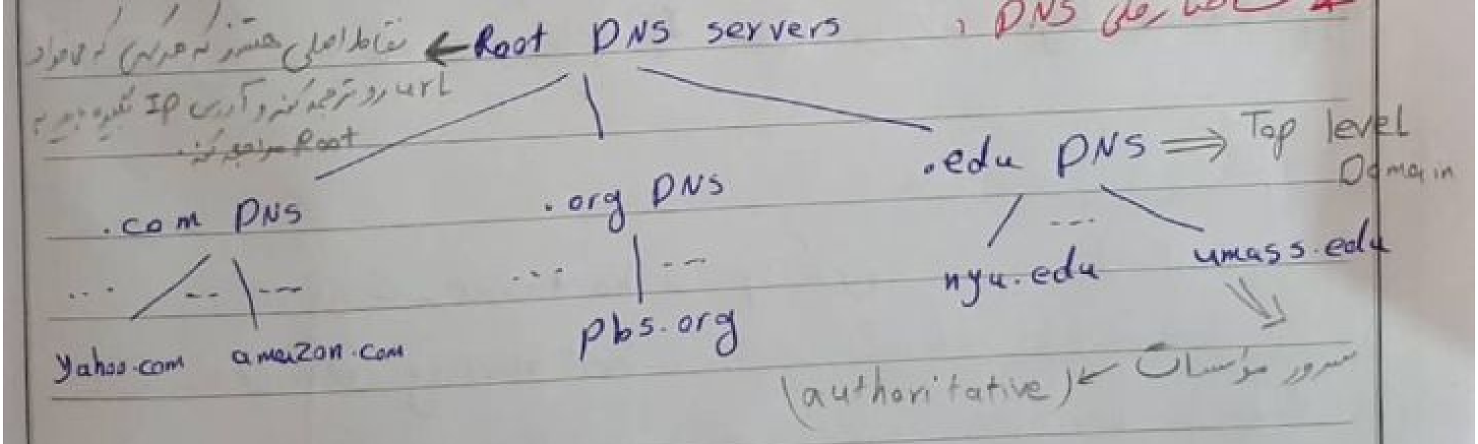
Year. Month. Date. ( ) Subject :

* چرا نباید DNS ← centralize نباشد؟

- چون اگه یدونه از اینها قطع بشه، کل شبکه قطع خواهد شد.
- traffic volume زیاد به‌سمتش میره اگه تنها باشه.
- در maintenance هم مشکل پیش میاد یعنی وقتی که براش مشکلی پیش اومد backup وجود نداره که تا زمانی که دارن مشکلشو برطرف می‌کنن از اون بشه استفاده کرد.

- single point of failure
- traffic volume
- distant centralized database
- maintenance

* ساختار کلی DNS



عملاً ۱۳ تا Root server در جهان وجود داره. که مدیریت اینها برای ICANN هست.

یکی از این ۱۳ تا رو خودش مدیریت می‌کنه و ۱۲ تای دیگه رو داده به افراد مختلف.

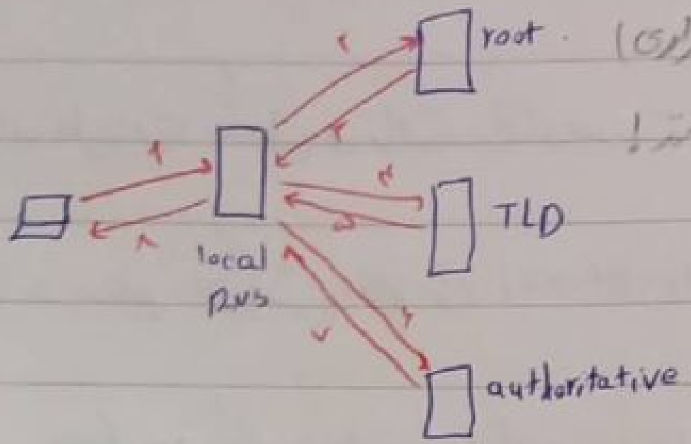
Year. Month. Date. ( ) Subject :

انواع درخواست از Root : ← iterated query (درخواست تکراری)

↙ recursive query

iterated query: (درخواست تکراری)

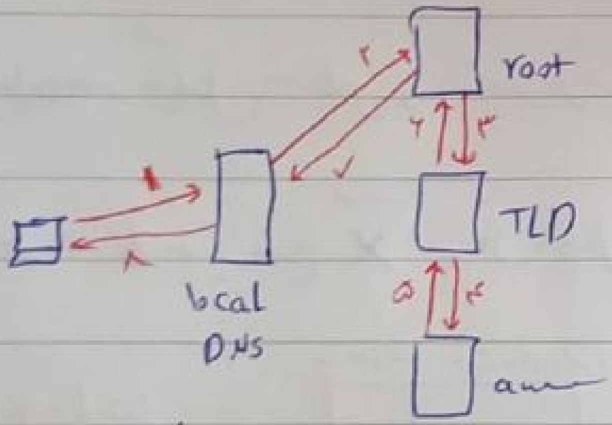
مشکل این روش: خیلی باید صحبت کنه!



recursive query :

سرورها باهم ارتباط دارند.

به صورت بازگشتی انجام می‌شود.



برای اینکه رفت‌وآمد زیاد نشه بهتره که DNS به شکل یه cache عمل کنه.

مزایای این کار

① response time رو پایین میاره.

② یه TTL براش تعریف میشه.

③ حتی TLD سرورها هم می‌تونن cache بشوند. که دیگه به root کار نداشته باشند.

* نحوه ذخیره سازی Data در Database DNS ؟

resource records

RR format: (name, value, type, ttl)

type

- A → [illegible] ← تبدیل url به Ip
- NS → [illegible] Domain DNS
- CNAME → [illegible] url
- MX

در امروزه استفاده از ویدیو بسیار زیاد شده است و اسا حجم زیادی رو به خودش اختصاص میده. و ممکنه هنگام دانلود ویدیو به مشکل برخورد کنیم.

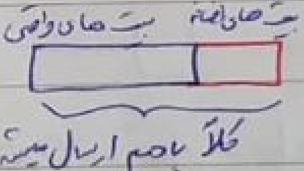
برای حل این مشکل می توان: ① به صورت distributed پیاده سازه شه.

② در سطح اپلیکیشن انجام شود (application_level infrastructure)

* video چگونه ارسال میشه (مثل تلویزیون)؟ همه image هام (۰و۱) دیجیتال هستن

برای هر پیکسل یه سطحی رو مشخص می کند مثلاً چه رنگی داره و ... برای مشخص کردن ۰_۲۵۵ نیاز به ۸ بیت داریم. $\overset{\text{ضرب}}{X}$ زمان ارسال

Coding :



- با این کار خطارا مشخص می کند آیا خطا رخ داده است یا نه؟
- تصحیح خطا هم انجام می دهد.

کانال coding

اما برای ارسال video از coding برای این استفاده می شود که بتوانیم video رو با نرخ پایین تری بتوانیم ارسال کنیم. عموماً به این نوع coding ← sorce coding گفته می شود.

تفاوت کانال coding و sorce coding: در کانال یه چیزی اضافه میشه به اطلاعات ولی هدف در sorce این هست که اطلاعاتی که ارسال می کنیم رو کمتر کنیم و بیت کمتری ارسال کنیم.

مثلاً می‌توانیم برای ارسال (یه فریم عکس (یک فریم)، می‌توانیم بگوییم از یه پیکسل تا فلان

spatial coding

پیکسل چه رنگی هست و ... که حجم کمتری استفاده می‌شود) (بعد برای فریم بعدی جاهایی

temporal coding

که تغییر می‌کنند را ارسال می‌کنیم.)

نرخ‌های مختلف برای ارسال:

- mpEG 1 (CD-Rom) 1.5 mbps
- mpEG 2 (DVD) 3-6 mbps
- mpEG 4 (often used in internet 64 kbps - 12 mbps)