The Domain Name System(DNS)

- همون طور که همه ی ما نام و کد ملی و شماره ی پاسپورت ... مخصوص به خودمون داریم، در شبکه های کامپیوتری هم host ها و end-system ها ، هم با نام شناسایی میشن (برای استفاده ی راحت تر کاربر های انسانی) و هم توسط IP address ِ ۳۲ بیتی که برای روتر ها و مسیریابی ازش استفاده میشه.
- DNS یه دیتابیس توزیع شده (distributed database) هست که شبیه یه دایرکتوری یا یه دفترچه تلفنه، در واقع اون نام و IP address رو به هم map می کنه.(چون این دیتابیس تعداد زیادی hostو end system داره ،به شکل توزیع شده پیاده سازی میشه)
- قسمت های مختلف این دیتابیس توی سرور های مختلفی هست و به این سرور ها ، name server گفته میشه.
- در کنار این سرور ها ، احتیاج به یه application-layer protocol داریم ، که همون DNS هست و host ها با DNS server ها ارتباط برقرار می کنن، و می تونن IP address متناظر با یه name رو پیدا کنن.
- Map کردن نام یه سرور (domain name) به یه IP address ،به صورت اولیه مربوط به core شبکه ست، چون روتر های core اینترنت

هستن که مسیریابی رو انجام میدن ، اما به خاطر گستردگی زیاد روتر ها و در دسترس نبودنشون ، تغییر توی core اینترنت خیلی پیچیدس . برای حل این مشکل، بدون اینکه تغییری توی ساختار روتر ها بدن، و این سرویس رو توسط سرور هایی که در لبه ی شبکه هستن و application layer هایی مثل DNS ، ارائه می کنیم و IP address رو از طریق کوئری هایی که با سرور ها انجام میدیم ، به دست میاریم و روتر ها می تونن بسته رو به مقصد برسونن.

- سرویس های DNS :

۱-hostname رو به IP-address تبدیل می کنه.

۲-host aliasing (نام شهرت) : برای یه host میتونیم چندین نام داشته باشیم. ممکنه تکنسین های موسسه ای که با یه سرور سروکار دارن ، اسم طولانی ای برای اون سرور انتخاب کرده باشن . ولی از دید بیرونی ، برای کاربرانی که میخوان با اون سرور در ارتباط باشن ، نیازی به جزئیات اون اسم نباشه و از آدرس کوتاه تری استفاده می کنن. به اسم اصلی سرور canonical name و به اسم های فرعی برای استفاده ی کاربران عادی، alias name گفته میشه.

DNS از این موضوع پشتیبانی می کنه و اگه نام های مختلف یه سرور رو ازش بپرسیم ، (چه canonical چه alias) همون IP address یکتا رو بهمون برمی گردونه. به این ترتیب می تونیم از مزایایی که

canonical name یا alias name دارن بهره مند بشیم و مشکلی ایجاد نشه.

- همچنین می تونیم یک اسم و چندین host داشته باشیم.

مثلا بعضی از شرکت ها هستن که هم یه web server دارن و هم یه mail server که جفت این ها alias name یکسانی دارن .

مثل یاهو که می تونیم www.yahoo.com تایپ کنیم و browser ما می فهمه که وب سرور این شرکت مد نظر ما هست .یا اینکه می خوایم از اپلیکیشن ایمیل یاهو استفاده کنیم ، در این صورت پروتکل SMTP که برای اپلیکیشن ایمیل استفاده میشه ، از سرور های DNS درخواست می کنه که IP address ِmail server های یاهو رو بهش بدن. بنابراین mail server و web server یاهو ، یک اسم دارن(yahoo.com) و IP address های این دو رو می تونیم با استفاده از درخواست هایی که از سرور های DNS می کنیم، تفکیک کنیم.

یه مثال دیگه ، شرکت هایی هستن که دارن وب اپلیکیشن های مهمی رو دارن سرویس میدن ، مثل گوگل ، فیس بوک ، ... و این ها برای اینکه بتونن به تعداد زیادی کاربر سرویس بدن، تنها یک سرور ندارن، بلکه replicated Web servers دارن، یعنی تعداد زیادی سرور که در داخل دیتا سنتر هایی در نقاط مختلف دنیا که IP address های مختلفی هم دارن. اما مثلا وقتی راجب IP address گوگل از سرور

های DNS کوئری می کنیم، میاد از مجموعه ی IP address هایی که در دست داره، یه تعدادی رو انتخاب می کنه و برای ما میفرسته. پس ما لازم نیست که نام تک تک سرور ها رو بلد باشیم ، همین که google.com تایپ می کنیم، یک نام هست و برای چندین سرور ارسال میشه و بعد DNS ، IP address های اون سرور ها رو برای ما میفرسته.

۳-متناظر با replicated Web server ، کار دیگه ای که DNS انجام میده ، load distribution هست.

مثلا تعداد سرور هایی که سرویس google.com رو دارن ارائه میدن میتونه زیاد باشه و هرکدوم یه IP address ای دارن. وقتی کاربرها میان راجب IP address این سرویس سوال می پرسن ، پاسخی که ارسال میشه ، میتونه یا IP address تصادفی از بین IP های موجود باشه.

چون DNS به طور رندوم انتخاب می کنه که چه IP ای اول باشه چه IP ای دوم و ... ، اگه فرض کنیم browser ها ،اولین IP رو انتخاب می کنن، این باعث میشه http request هایی که توسط کاربرهای مختلف به سرور های گوگل ارسال میشه ، بینشون پخش بشه و DNS به نحوی load balancing یا load distribution انجام داده.

- به طور کلی DNS نقش پر رنگی توی شبکه ی اینترنت ایفا می کنه و اگه از کار بیفته ، اینترنت هم کارکردی نداره!

- DNS توسط سرور های زیادی ارائه میشه و توزیع شده ست. سوالی که پیش میاد اینه که چرا از یه سرور قدرتمند به صورت centralized استفاده نمی کنیم؟

۱-single point of failure :اگه کل سرویس DNS توسط یه سرور ارائه بشه ، و اون سرور مشکلی براش پیش بیاد ، DNS هم از کار میفته و با توجه به نقش حیاتی ای که داره، کل کارکرد اینترنت هم دچار مشکل میشه.

۲-traffic volume : شاید میلیارد ها DNS query در روز انجام بشه و چون DNS یه اپلیکیشن basic هست، و بسیاری از اپلیکیشن ها برای اینکه کارکردی داشته باشن در لایه های پنهان خودشون ابتدا از DNS استفاده می کنن تا host name رو به IP address ، map کنن و بعد تازه میتونن سرویس خودشون رو شروع کنن.
پس تعداد زیادی DNS query انجام میشه و اگه فقط یه سرور داشته باشیم ، به دلیل حجم بالای درخواست ها ، دچار مشکل میشه.

۳-distant centralized database : مجددا به خاطر اینکه DNS یه اپلیکیشن basic هست، اگه بخوایم به صورت متمرکز این سرویس رو ارائه بدیم ، چون فاصله ی سرور DNS از جاهای مختلف ، متفاوته ، در فاصله های زیاد، برای کاربران مشکل به وجود میاد و تاخیر mapping در عملکرد شبکه تاثیر میذاره.

۴- maintenance : چون دیتابیس خیلی بزرگه و داینامیک خیلی زیادی هم توی رکورد های دیتابیس شاهد هستیم و تغییرات قابل توجهی انجام میشه، (مثلا host های جدیدی به دیتابیس اضافه بشن یا نامشون تغییر بکنه) اینکه بتونیم این دیتابیس رو توسط یک سرور ، در یک جا مدیریت کنیم ، کار خیلی سختی هست.

پس به طور کلی ، روش متمرکز شدن ، قابلیت مقیاس پذیری (scalable) نداره.

- حالا سوال اول که پیش میاد اینه که بخش های مختلف این دیتابیس در چه سرور هایی ذخیره بشن؟
- سوال دومی که پیش میاد اینه که اگه بخوایم از سروری سوال بپرسیم، از کدوم بپرسیم؟ یعنی به عنوان یه کلاینت، اون IP address متناظر با host name ای که داریم دنبالش می گردیم، رو روی کدوم سرور پیدا کنیم؟

این سوال ها مربوط میشن به ساختار سلسله مراتبی DNS و این که این سرور ها باید در قالب یه ساختار سلسله مراتبی(hierarchical) ، روابطشون تنظیم بشه.

سروری که در سطح اول هست، Root نام داره.

سرورهایی که در سطح دوم قرار دارن Top Level Domain (TLD) نام دارن.

سرور های سطح سوم هم Authoritative نام دارن.

رکورد هایی که دنبالشون می گردیم در سرور های Authoritative قرار دارن.

سرور های سطح یک و دو هم با ما میگن رکورد هایی که دنبالشون می گردیم توی کدوم یک از سرور های Authoritative قرار گرفته.

مثال : یه کلاینت می خواد IP address سایت www.amazon.com رو پیدا کنه . تنها اطلاعاتی هم که داره آدرس Root DNS Server هست. ابتدا از همین سرورهای Root DNS سوال می کنه ، و این سرور به آدرس Domain Name نگاه می کنه که .com هست. بعضی از سرور های TLD که در سطح دوم قرار دارن، مسئول Domain Name های .com هستن که بهشون میگن .com DNS servers و بعد از اون ، سرور Root ، میاد آدرس یکی از سرور هایی که مسئول Domain Name های .com هست رو به کلاینت بر می گردونه و کلاینت میره سوالش رو از اون سرور می پرسه. حالا این سرور، آدرس سرور Authoritative سایت آمازون که IP address این سرویس داخلش هست رو می دونه و این آدرس رو برای کلاینت ارسال می کنه.

- **Root name servers**

- این سرور ها در سطح اول DNS قرار دارن و اصطلاحا بهشون contact-of-last-resort گفته میشه ، یعنی اگه هیچ اطلاعی راجع به سرور های DNS نداریم ، می تونیم به سرور های Root مراجعه کنیم و طبق روندی که توی قسمت قبل توضیح داده شد، به IP address ای که دنبالش هستیم دسترسی پیدا کنیم.
البته در خیلی از موارد ما آدرس TLD ها رو توی browser cache یا Local DNS server ذخیره شده و از اول درخت شروع نمی کنیم و شورتکات می زنیم.
- تعداد DNS server ها حدود ۱۳۰۰ تا هست که در کشور های مختلف پراکنده شدن. از بابت نام و IP address ، اون ها رو در قالب ۱۳ نام (۱۳ تا ورژن ۴ و ۱۳ تا ورژن ۶) نام گذاری می کنیم.
یعنی هر ۱۰۰ تا از این سرور ها ، یک نام و یک IP address دارن.
- از طرفی این دسته های مختلف توسط شرکت های مختلف مدیریت میشن که تعداد این شرکت ها هم ۱۲ تا هست. اون موسسه ای که وظیفه ی هماهنگی بین همه ی این شرکت ها و وضع قوانین رو به عهده داره ، ICANN(Internet Corporation for Assigned Names and Numbers) هست.

- شرکت IANA شرکت مادری هست که ICANN ازش جدا شده و وظیفه ی هماهنگی رو به عهده گرفته.(وب سایت: iana.org)
توی این وب سایت ، اومدن راجع به root server ها و دسته های مختلفشون اطلاعاتی رو بیان کرده.
نام این ۱۳ تا سرور مختلف ، شامل یک حرف و بعد «.root-server.net» هست . بعد از نام ، IP address و اون اوپراتوری که این سرور ها رو مدیریت می کنه ذکر شده.
- برای داشتن یه DNS server، حداقل اطلاعات مورد نیاز برای اینکه کلاینت ها از این سرور ها استفاده کنن، اینه که یکی از این آدرس های root server رو داشته باشیم تا بتونیم به آدرس سرورهای Authoritative برسیم.
- Browser ها یا local DNS server ها ، اون اطلاعاتی که از DNS به دست میارن رو برای مدتی توی cache خودشون ذخیره می کنن تا اگه کلاینت از اطلاعاتی که قبلا درخواست کرده ، یا توی local DNS server کلاینت های دیگه درخواست کردن، بتونه به این اطلاعات سریع دسترسی داشته باشه و متحمل تاخیر پرسش توی سلسله مراتب DNS نشه.
- یه وب سایت دیگه هست به اسم root-server.org که اطلاعات مربوط به root server ها رو آورده.

- **TLD (Top-Level-Domain)**

- Domain Name هایی که استفاده می کنیم معمولا پسوند های متفاوتی دارن، مثل .com ، .org ، .net ، .edu و ... یا مثلا متناظر با کشور های مختلف پسوند های مختلف داریم مثل .cn ، .uk ، .fr ، .ir و...

متناظر با هرکدوم از این پسوند ها ، TLD متناظر با اون پسوند رو داریم و یه سری سرور های به خصوص هم هستن که این پسوند ها رو پشتیبانی می کنن. مثل .com DNS server ، .org DNS server و ....

- این سرویس های TLD، معمولا توسط شرکت های خصوصی مدیریت میشن .

از معروف ترین این شرکت ها ، Network Solutions رو میشه نام برد که از شرکت های قدیمی ایه که TLD های .com و .net رو مدیریت می کنه و مالک این سرور ها هست.

شرکت Educause هم edu TLD رو مدیریت می کنه.

- **Authoritative DNS servers**

- سرور هایی هستن که عمدتا توسط سازمان های مختلف خصوصی مدیریت میشن و IP address هایی که دنبالشون هستیم در این سرور ها قرار دارن.
- بعضی از موسسات یا خودشون این سرور ها رو دارن یا از Authoritative DNS servers هایی که ISP هاشون دارن، استفاده می کنن.

- **Local DNS name servers**

- وجود این سرور ها الزامی نیست ، بلکه کمک کننده هست.
- موسسه ها یا ISP هایی که Local DNS servers دارن ، کلاینت هاشون به جای اینکه مستقیما از DNS servers سوال بپرسن ، از Local DNS server ها می پرسن و این سرورهای لوکال ، یا پاسخ سوال رو به نمایندگی از کلاینت ها ، از DNS server ها می پرسن، یا پاسخ سوال رو توی cache خودشون دارن. به این ترتیب از تاخیر رسیدن جواب ها از DNS جلوگیری می کنه و سادگی کار برای کلاینت ها بیشتر میشه.

- با استفاده از این دستور ها میشه نام local DNS server ها رو پیدا کرد.

- each ISP has local DNS name server; to find yours:
  - MacOS: scutil –dns
  - Ubuntu: nmcli device show <interface_name>
  - Windows: ipconfig /all

این اطلاعات توسط پروتکل DHCP به دست میاد.

- **DNS name resolution: iterated query**

- هر وقت که ما می خوایم یه IP address رو انتخاب کنیم، سرور هایی که توی level های مختلف DNS هستن، نمیان جواب های نهایی رو به local DNS server یا کلاینت بدن، بلکه مرحله به مرحله آدرس سرور های پایین تر رو میدن و خود اون ها اون مسئولیت رو دارن که برن تک تک از سرور ها آدرس سرور های پایین تر رو بپرسن تا در نهایت IP address رو از یه authoritative server بپرسن.

- **DNS name resolution: recursive query**

- توی این روش وقتی یه سوالی از یه سروری می پرسیم ، به یه سرور دیگه ارجاع داده نمیشیم که سوال رو از اون بپرسیم ، بلکه جواب نهایی رو از همون سرور دریافت می کنیم. یعنی اول از local DNS server می پرسیم، local DNS server از root ، root از TLD و TLD از authoritative می پرسه و این فرآیند برای ارسال پاسخ به طور برعکس تکرار میشه و نهایتا به کلاینت می رسه.
- تفاوتی که این روش با روش قبلی داره که باعث میشه ازش خیلی استفاده نکنن ، اینه که load زیادی رو روی سرور های سطح بالاتر ایجاد می کنه. چون تعداد کوئری های سرور های DNS خیلی زیاده و اگه هر root DNS server ای بخواد این حالت رو در خودش حفظ کنه که منتظر دریافت پاسخ نهایی از سرور سطح بالاتر باشه ، و بعد جواب یه کوئری رو بده ، load اش خیلی زیاد میشه و هرچی سطح بالاتر بره این load بیشتر هم میشه.

- البته این مثال ها راجب iterative و recursive ، در حالتی هستن که هیچ اطلاعاتی داخل local DNS server ، cache نشده .

اگه IP address مربوط به host name ای که دنبالش هستیم داخل

local DNS وجود داشت و این اطلاعات کهنه نبودن، همون موقع بدون سوال پرسیدن این اطلاعات رو برای کلاینت می فرستاد .

گاهی هم IP address رو نداره ، ولی TLD مربوط به پسوند domain name رو داره ، می تونه شورتکات بزنه و سوال رو از TLD DNS server بپرسه.

- **Caching DNS Information**

- به جز local DNS ها ، سرور های دیگه مثل TLD هم caching انجام میدن. البته عمده ی کار caching توی سرور های local DNS انجام میشه.
- این فرایند caching میتونه باعث صرفه جویی توی زمان بشه ، ولی مسئله ای که پیش میاد اینه که هرجا caching مطرحه ، به روز بودن اطلاعات caching هم مطرحه. توی DNS server ها این قضیه این طور مطرح میشه که همراه با بقیه ی اطلاعاتی که داخل این رکورد ها هست ، مثل IP، host name و ... یه فیلد TTL(Time To Live) هم داریم که زمان منقضی شدن اون رکورد رو به سرور local DNS اعلام می کنه.

بنابراین اگه قبل از زمان TTL ، یه کوئری به DNS برسه، مستقیم از نسخه ی درون خودش استفاده می کنه و جواب کلاینت رو ارسال می

کنه . اگرهم بعد از زمان TTL شد ، که مثل اولین بار که caching انجام میشه ، باید اطلاعات رو به روز کنه .

- اگه یه host name ای ، IP address اش رو عوض کنه ، تا زمانی که تمام TTL ها توی local DNS server منقضی نشن، این تغییرات مشهود نمیشن . به خاطر همینه که مثلا اگه یه web page ای رو راه اندازی کنیم ، گفته میشه که تا حدود ۲۴ ساعت بعد از راه اندازی، ممکنه همه جا در دسترس نباشه.
- بنابراین این سرویس name-to-address translation(سرویس DNS) رو میگن best-effort هست. یعنی ضمانتی راجع به به روز بودن یا تاخیر نداشتن دریافت IP address به ما نمیده.