PROJECT 1

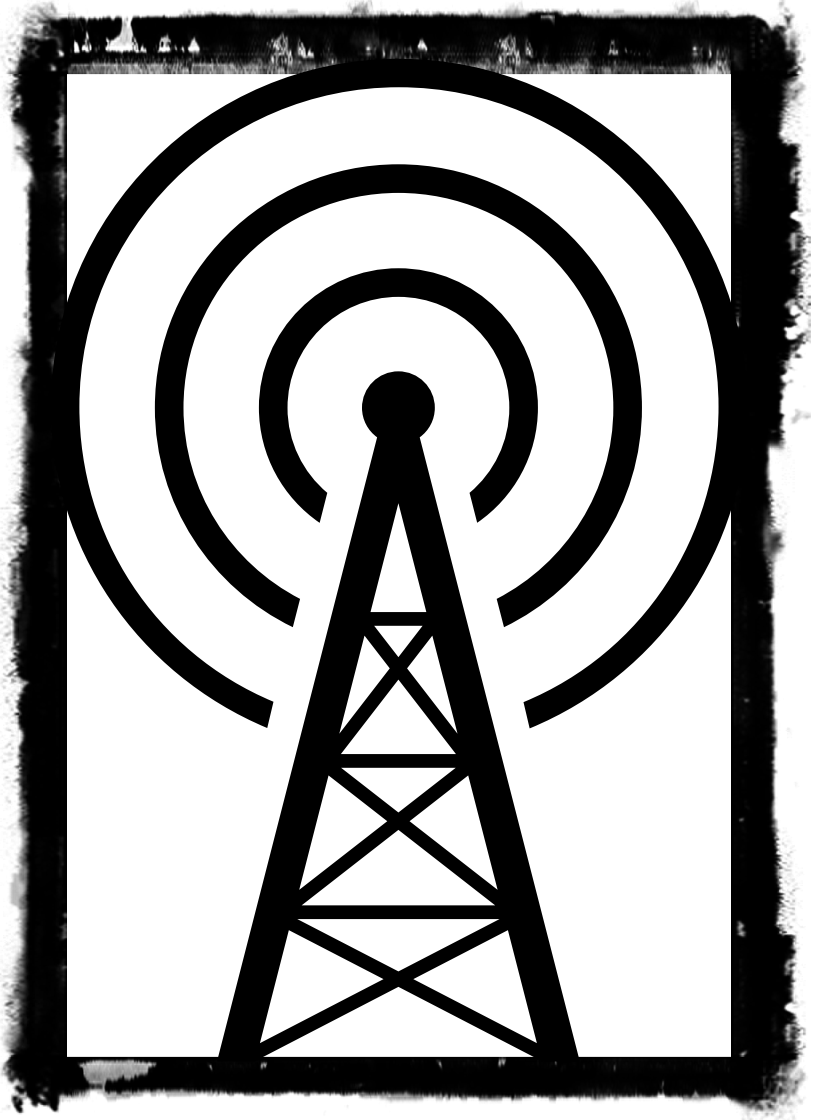
COMPUTER NETWORKS

BY MAHAN AHMADVAND

PROJECT 1

COMPUTER NETWORKS

BY MAHAN AHMADVAND



PART 1

۱) از پروتکل DNS چه استفاده ای می شود؟

برخی از سرویس ها از دید کاربران پنهان هستند و به اصطلاح به صورت host by host به کاربران ارائه می شوند، مانند سرویس DNS.

کار اصلی سرویس DNS ایجاد تناظر بین نام ها و آدرس های سیستم ها است، هر سیستم یک نام دارد(Host name) و یک آدرس دارد که به آن IP address می گوییم.

چون آدرس ها دارای طول و ساختار ثابت هستند، پردازش آنها برای تجهیزات ممکن است و راحت تر است، اما کارکردن با آدرس ها برای انسان ها مشکل است و انسان ها با Host name راحت تر هستند و از طرفی نام ها جنبه های مفهومی هر سیستم را نیز شامل می شوند، حال تجهیزات از آدرس ها استفاده می کنند و ما انسان ها از نام ها، حال باید این دو را باهم Match کنیم، که این کار را سرویس DNS برای ما انجام می دهد(کار DNS تبدیل نام ها به آدرس های IP است).

در قدیم برای نام گذاری سیستم ها از یک ساختار تک سطحی استفاده می شد و هیچ سلسه مراتبی در نام گذاری وجود نداشت و مدیریت این نام ها از طریق فایل ای به نام Host انجام می شد، یعنی اگر سیستمی می خواست به بستر شبکه متصل شود باید نام خود را برای یک مرکز مدیریت فایل Host ایمیل می کرد و آن مرکز فایل Host تمام کامپیوتر ها را

آپدیت می کرد و برای آن ها ایمیل ارسال می کرد و آنها به روزرسانی را در سیستم خود انجام می دادند تا آن سیستم جدید می توانست به بستر شبکه متصل شود یک رول بسیار زمان بر بود، امروزه این روش به عنوان یک روش جایگزین برای سرویس DNS وجود دارد، یکی از ایرادات این روش زمان بر بودن آن، و ایراد دیگر آن آن بود که نام ها ساختار تک سطحی داشتند و به هرسیستمی یک نام یکتا اختصاص داده می شد، با بیشتر شدن تعداد سیستم های متصل بر بستر اینترنت این ساختار تک سطحی دیگر جواب نمی داد و مجبور شدند یک عدد تصادفی به ابتدای این نام ها اضافه کنند تا همچنان بتوانند این نام ها را یکتا نگه دارند، اما این مدل به دلیل پیچیدگی کنار گذاشته شد، سپس ساختار تک سطحی را کنار گذاشتند و از ساختار سلسه مراتبی ایجاد کردند، در این ساختار سلسه مراتبی فضای اینترنت به مجموعه ای از دامنه ها تقسیم شد و یکتایی نام فقط در داخل دامنه تضمین می شد مثلا ما دامنه های google.com و facebook.com داریم، اما نام www در هردو مشترک است www.google.com و www.facebook.com.

پس نام ها تبدیل شدند به نام های تحت دامنه و سیستمی که برای مدیریت این نام ها به وجود آمد اسم آن را DNS گذاشتند.

DNS = Domain Name System

سرور های DNS به صورت توزیع شده هستند و نسبت به هم یک ساختار سلسه مراتبی دارند.

دو نقش در سرویس DNS وجود دارند :

Reslover :

نقش client را بازی می کند و برروی سیستم ها نصب است و وظیفه ی آن query زدن و ارسال در خواست برای name server ها است.

Name server :

در واقع کار تبدیل نام ها و آدرس ها را انجام می دهد و به درخواست های Reslover و query های آن پاسخ می دهد.

پس تا به اینجا فهمیدیم که خدمت اصلی DNS تبدیل نام ها به آدرس ها است، در کنار این می توانیم در سرویس DNS، Canonical name داشته باشیم که به آن CNAME نیز می گوییم، مثلا یکی از پرکاربردترین این CNAME ها www است، که درواقع برای تمامی دامنه های webserver ها از نام مستعار(alias names)، www استفاده می کنند یعنی ممکن است نام آن webserver پیچیده باشد، اما با این نام مستعار این پیچیدگی را از بین برده ایم، همچنین برای ایمیل نیز نام مستعار داریم که دیگر به آن نمی پردازیم.

یکی از کارهای دیگری که سرویس DNS انجام می دهد، تعادل بار یا Load Balancing است، یعنی درخواست ها را بین webserver ها تقسیم می کند مثلا یک سرور گوگل در آمریکا وجود دارد و یکی دیگر در کانادا و DNS تقسیم درخواست بین این دو سرور گوگل انجام می دهد، مدل تقسیم باری که بین سرور ها با یک نام مورد استفاده قرار می گیرد، مدل round robin است، یعنی درخواست ها یکی درمیان بین این سرور ها تقسیم می شوند درخواست اول برای سرور اول، در خواست دوم برای سرور دوم و بعد دوباره برمی گردد به حالت اول.

همچنین تکنیک دیگری در کنار round robin وجود دارد به نام تکنیک netmask ordering در این تکنیک در واقع در

خواست client را به نزدیک ترین سرور از نظر موقعیت جغرافیایی ارجاع می دهد.

حال به برخی از اهداف طراحی DNS می پردازیم :

۱) بزرگ بودن پایگاه داده ها و آپدیت های پی در پی پایگاه داده ها نیازمند این هستند که در واقع این پایگاه های داده بصورت توزیع شده نگه داری شوند و نیازمند local caching هستند تا عملکرد بهتری داشته باشند و نگه داشتن کپی از پایگاه های داده سخت و هزینه بر هستند پس باید از این کار پیشگیری کرد.

۲) trade off بین هزینه ی بدست آوردن داده، سرعت آپدیت داده و دقت cache ها را باید منبع هندل کند.

۳) هزینه های پیاده سازی چنین تکنولوژی ای نشان می دهد که فقط وابسته به یک application خاص نیست.

ما باید بتوانیم با استفاده از نام ها آدرس ها، داده های mailbox و ... را دریافت کنیم.

۲) رکوردهای مختلف DNS را نام ببرید و هر یک را به صورت مختصر توضیح دهید.

اطلاعات مربوط به سرویس DNS در قالب resource records یا به صورت مخفف RR در دیتابیس سرویس DNS ذخیره می شود، این RR ها شامل چهار فیلد هستند، قسمت اول نام رکورد را مشخص می کند مانند یک کلید عمل می کند برای پرسش، حال آن کلید یک مقدار نیز دارد و از طریق مقدار می توان به محتوای کلید دسترسی پیدا کرد، مرحله ی بعدی type است که نوع رکورد را مشخص می کند و در نهایت ttl را داریم که زمان انقضای رکورد را تعیین می کند.

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

type ها در واقع مقادیر name و value را در RR ها مشخص می کنند.

حال هریک از type ها را توضیح می دهیم.

type = A :

یکی از پرکاربردترین type ها در سرویس DNS است، در واقع یک Host را از طریق نام معادل سازی می کند با یک آدرس IP، یعنی hostname مربوط به یک Host در قسمت name قرار می گیرد و مقدار آدرس IP آن در قسمت value قرار می گیرد، یک تناظر بین نام و آدرس IP از طریق رکورد A انجام می شود.

همچنین ما می توانیم رکورد های AAAA نیز داشته باشیم، در واقع تفاوت این رکوردها با A این است که یک hostname را متناظر می کنند با یک آدرس IP از نوع ورژن 6(ما برای

پروتکل IP در بستر اینترنت در حال حاضر دو نسخه داریم IP ورژن 4 و IP ورژن 6).

آدرس های IP ورژن 4، آدرس ها 32 بیت هستند، اما آدرس های IP ورژن 6، آدرس های 128 بیت هستند.

type = NS :

زمانی که این type را داریم، در واقع رکورد DNS برای ما name server را مشخص می کند، در قسمت name دامنه قرار می گیرد(می خواهد بگوید یک name server مسئول کدام دامنه است) و در قسمت value، hostname مربوط به name server را قرار می دهد.

معمولا اطلاعات مربوط به سرویس DNS که برای مدیریت یک دامنه مورد استفاده قرار می گیرد در پایگاه داده ای ذخیره می شود به اسم zone file، این zone file روی name server ای که در رکورد NS تعریف کرده ایم وجود دارد و از طریق این رکورد داریم مشخص می کنیم که اطلاعات مربوط به zone file از روی چه سروری قابل برداشت است.

حال برای هر name server در پایگاه داده ی DNS دو رکورد داریم، یک رکورد A برای اینکه hostname را با آدرس IP متناظر کنیم و یک رکورد NS که دامنه را متناظر با hostname می کند.

type = CNAME(canonical name) :

از طریق canonical name یک نام مستعار را متناظر می کنیم با یک hostname، یعنی در قسمت name نام مستعار قرار می گیرد مثلا www و در قسمت value، canonical domain name قرار می گیرد یعنی مثلا :

CNAME www ns1.aut.ac.ir

type = MX :

برای مشخص کردن mail server است و خود کلمه MX از mail exchanger می آید و در واقع hostname یک mail server را که در قسمت value قرار می گیرد را با نام درنظر گرفته شده برای اون mail server متناظر می کند، همچنین به این نکته توجه شود که می توانیم اولویت برای mail server ها نیز درنظر بگیریم یعنی اگر چندین mail server داریم براساس اولویت اشون request ها بهشون ارجاع داده بشود.

رکورد MAILA یک رکورد واقعی نیست، اما یک query type است که رکوردهای MF و MD را برمیگرداند و این رکوردها امروزه با تایپ MX جایگزین شده اند این دو گزینه برای آدرس های پست الکترونیکی کاربرد دارند و دیگر استفاده نمی شوند.

MD(mail destination)

MF(mail forwarder)

رکوردهای MB و MG و MR و MINFO، رکوردهایی برای منتشر کردن subscriber mailing list ها هستند.

با رکورد MINFO میتوان اطلاعاتی در مورد سیستم پست الکترونیکیِ متناظر با یک حوزه ارائه نمود

MAILB یک query code است که یکی از رکورد های گفته شده را برمی گرداند.

هدف MB و MG آن بود که دستور های SMTP VRFY و EXPN را عوض کنند.

MR بـرای عـوض کـردن "SMTP 551 User Not Local" error بود.

بـعدا RFC 2505 اعـلام کـرد کـه هـردو VRFY و EXPN از کار بیفتند و این باعث شد که MB و MG بی اهمیت شوند.

همچنین با MG مدیر مسئول شبکه میتواند افراد حقیقی را به عنوان دارندهٔ صندوق پستی در یک حوزه تعریف نماید.

TXT :

Text Record برای اضافه کردن هرگونه توضیح به کار می رود، همچنین Text Record میتواند برای سیستم تصدیق ایمیل spf و همچنین به منظور دادن و فراهم آوری اطلاعات مربوط به آن به کار رود.

WKS :

مخفف Well Known Services و نمایانگر سرویس های معروف موجود در ناحیه که البته با وجود رکوردهای SRV نیازی به تعریف این رکورد نیست.

در واقع این برای توصیف Well Known Services و با استفاده از یک host، ساپورت می شود. در عمل استفاده نمی شود. توصیه و عمل فعلی این است که آیا یک سرویس برروی یک آدرس IP ساپورت می شود یا خیر.

حتی SMTP نیز ممنوع کرده از استفاده از رکورد WKS در پردازش MX.

SOA :

این رکورد هم مانند NS Record، اطلاعاتی درباره‌ی DNS zone می‌دهد. اطلاعاتی مانند DNS معتبر این zone، اطلاعات تماس admin دامنه، شماره سریال دامنه و ... در این رکورد نگهداری می شوند. این رکورد دارای بخش های متفاوت است که در زیر توضیح داده می شود :

negative TTL : اگر سرور نتوانست نام مورد درخواست را در تایم در نظر گرفته شده ارسال نماید خطای کش نشان می دهد.

expiry period : اگر یک DNS سرور نتواند در زمان درج شده در فاکتور مذکور، دیتابیس خود را بروز نماید و از سایر سرور ها نیز جوابی دریافت ننمود، درخواست را لغو شده در نظر می گیرد.

retry interval : به زمان رفرش اطلاق می شود. در واقع هر زمان DNS سرور نتواند پاسخ درخواست را در تایم مشخص شده ارسال نماید، بر اساس زمانی که در این بخش ذکر شده مجدد درخواست را بروز می نماید.

Minimum ttl : به حداقل زمانی گفته می شود کهDNS SERVER درخواست ها را در حافظه کش خود نگهداری می نماید.

PTR :

PTR Record که یک رکورد DNS معکوس نامیده می‌شود، یک IP را به یک آدرس دامنه ارجاع می‌دهد. دقیقا برعکس همان کاری که رکورد A انجام می‌دهد. در واقع این رکورد یک اتصال

صحیح بین دامنه و آی پی برقرار می‌کند تا درخواستها اشتباها به سرورهای دیگر ارسال نشود.

سرورهای روی اینترنت از رکوردهای PTR که یک رکورد اختیاری است برای جمع آوری لاگ ها و اطلاعات، اطلاع‌رسانی در تصمیم گیری های درخواست اسپم و نمایش جزئیات استفاده می کنند. اغلب سرورهای ایمیل وقتی یک ایمیل جدید دریافت می‌کنند، آدرس IP رکورد PTR آن را بررسی می‌کنند. رکورد ذکر شده به عنوان تایید هویت درخواست ها استفاده می شود و چنانچه درخواستی فاقد این رکورد باشد به این درخواست به صورت اسپم برخورد می شود.

HINFO :

این رکورد اطلاعات در مورد CPU و سیستم عامل یک host می دهد. هدف این بود که به پروتکل ها اجازه دهد هنگام برقراری ارتباط با peer های مشابه دیگر پردازش را بهینه کنند.

همچنین مخفف **Host Information** و یک رکورد متنی اختیاری راجع به نوع سخت افزار و سیستم عامل بکار رفته در یک سرویس میزبانی وب است. به دلایل امنیتی در بیشتر سرورهای عمومی از چنین رکوردی استفاده نمی شود.

NULL :

استفاده از این گزینه رکورد را غیرعملیاتی و بی ارزش می نماید و برای زمانی کاربرد دارد که ترجیح داده شود بجای حذف ، رکوردی تبدیل به یک رکورد بی ارزش شود.

AXFR :

تـــمـام zone file را از master name server بـــه secondary name server ها منتقل می کند.

* :

هـمه ی رکـوردهـا از هـمه ی type هـای شـناخـته شـده بـرای name server را برمی گرداند.

اگر name server هیچ اطلاعاتی درمورد نام موردنظر ندارد، در خواست ارسال خواهد شد.

مـثلا اگـر هـم A و MX بـرای یـک نـام وجـود دارنـد امـا name server فـقط رکـورد A را بـه صـورت cache شـده داراسـت، فـقط رکورد A بر می گردد.

۳) DNS server چیست و آدرس سه مورد از معروف ترین DNS serverها را نام ببرید.

DNS دفترچه تلفن اینترنت است، هنگامی که کابران مثلا در مرورگر خود تایپ می کنند google.com، در واقع DNS مسئول پیدا کردن آدرس IP درست برای google.com است، سپس مرورگر از آدرس IP برای ارتباط با origin server ها و CDN edge server ها می کند تا بتواند به اطلاعات وب سایت ها دسترسی پیدا کند و این کار به دلیل وجود DNS server ها انجام می شود و در واقع ماشین هایی هستند برای جواب دادن به کوئری های DNS.

توجه شود که معماری مورد استفاده در سرویس DNS، client server است.

4 نوع DNS Server داریم :

1) Local nameservers
2) Root nameservers
3) TLD nameservers
4) authoritative nameservers

ابتدا باید با دو نوع نرم افزار در سرویس DNS آشنا شویم :

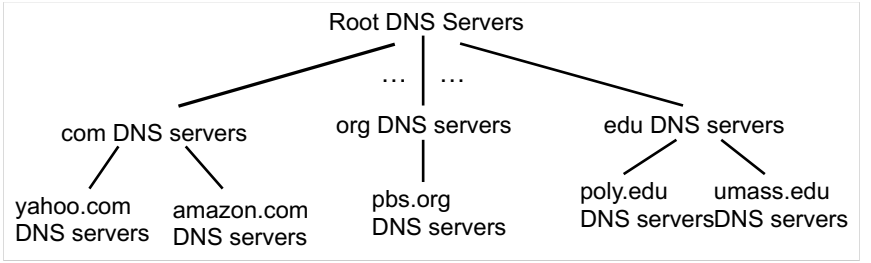
Reslover :

نقش client را بازی می کند و برروی سیستم ها نصب است و وظیفه ی آن query زدن و ارسال در خواست برای name server ها است.

Name server :

در واقع کار تبدیل نام ها و آدرس ها را انجام می دهد و به درخواست های Reslover و query های آن پاسخ می دهد.

حال هر یک از 4 نوع DNS Server را توضیح می دهیم :



شکل نشان داده شده ساختار سلسه مراتبی سرویس DNS را نشان می دهد، در بالاترین سطح Root nameservers ها قرار دارند.

در واقع مثلا زمانی که می گوییم www.google.com این یک دات در انتها نیز دارد که به root nameserver موردنظراش اشاره می کند .www.google.com(دات آخر root server) را مشخص می کند. حدودا 400 تا root server وجود دارد که مدیریت آنها توسط 13 سازمان انجام می شوند.

این root server ها اطلاعات مربوط به دامنه های زیر مجموعه ی خودشان را ذخیره می کنند یعنی سرور هایی که مسئول هر دامنه هستند را می شناسند و در واقع root server ها اطلاعاتی در مورد host ها ندارند و صرفا نحوه ی دسترسی به name server های دامنه ی زیرمجموعه ی خودشان را می دانند.

بعد از این root server ها، دامنه های سطح بالا را داریم که اصطلاحا به آنها Top Level Domain می گویند TLD nameservers، خود TLD ها می توانند gTLD باشند یا می توانند ccTLD باشند.

gTLD ها می شوند Generic TLD ها و ccTLD ها می شوند Country Code TLD.

ccTLD ها دامنه های سطح بالایی هستند که در اختیار دولت کشور ها قرار دارند، معمولا دو حرف از نام آن کشور را شامل می شوند مانند :ir، uk، us

و اینها در اختیار سازمان های حکومتی قرار دارند و به دولت ها اختصاص پیدا می کنند

gTLD ها به سازمان های تجاری اختصاص پیدا کرده اند مانند com(مخفف commercial)، edu(مخفف education)، org(مخفف organization) و ... .

این TLD ها شرکت هایی را دارند مسئول ثبت اطلاعات نام ها و زیر دامنه های مجموعه ی خودشان، و اگر دامنه ای بخواهیم، باید به این شرکت ها درخواست دهیم و یک دامنه به اسم خود ثبت کنیم، زمانی که نام خود را به یک TLD اعلام می کنیم باید آدرس DNS server خود را نیز بدهیم تا برای آن TLD مشخص شود چه سروری مسئول تبدیل نام ها به آدرس های IP است.

پس فهمیدیم که همچنین name server های TLD نیز اطلاعاتی درباره ی host ها ندارند و صرفا name server های که در سطح دامنه های TLD هستند را می شناسند.

اگر کاربری برای root name server ها در خواستی بفرستد root name server ها ارجاع می دهند به name server های TLD و آن ها نیز ارجاع می دهند به name server های زیرمجموعه ی خود.

سه نوع name server داریم : primary، secondary و cache-only.

Name server های primary اطلاعات مربوط به نام ها و آدرس های IP را در یک دامنه به صورت local دارند و روی دیسک محلی خود آنها را ذخیره کرده اند، Name server های secondary اینها نیز اطلاعات مربوط به نام ها و آدرس های IP را به صورت کامل در یک دامنه دارا هستند اما این اطلاعات را از روی دیسک خودشان نمی خوانند و بصورت دوره ای از روی Name server های primary آپدیت می شوند، یعنی لود را از روی Name server های primary بر می دارند.

یعنی بار را روی چندین Name server secondary ذخیره می کنیم.

اما name server های cache-only به صورت دوره ای آپدیت نمی شوند هر زمانی که request ای برایشان ارسال شود، این request را ارجاع می دهند به یک name server، primary یا secondary زمانی که پاسخ را دریافت کردند، در حافظه ی خود cache می کنند تا نیازی نباشد که در خواست های بعدی را ارجاع دهند و اینها در شبکه ی local، client قرار دارند.

حال authoritative nameserver ها کلیه اطلاعات نام ها و آدرس های IP را شامل می شوند یعنی می توان به این nameserver ها اتکا کرد حال می تواند یک nameserver، primary باشد یا secondary باشد و این nameserver ها در زمان ثبت دامنه به شرکت های ثبت کننده ی دامنه معرفی می شوند.

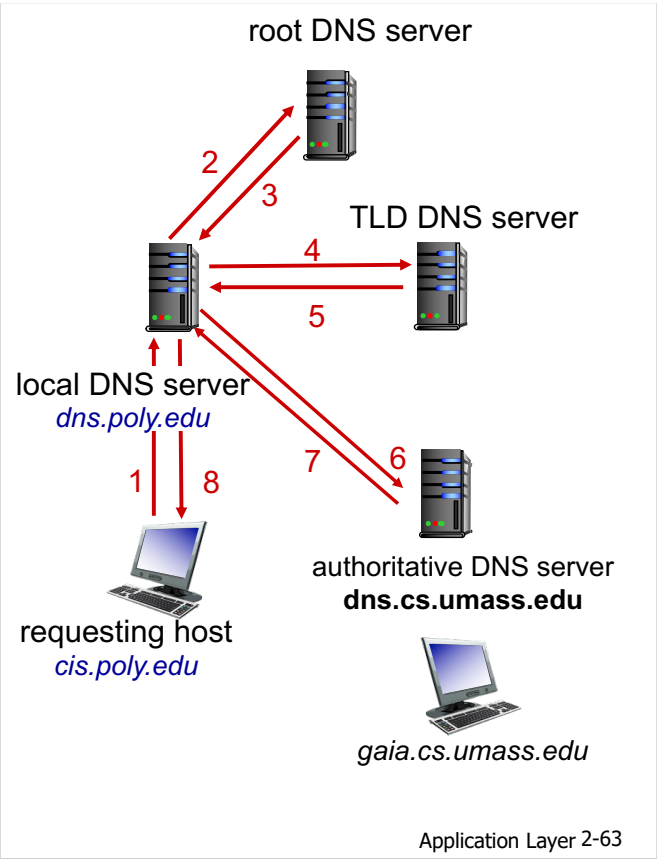
پس می فهمیم که non-authoritative nameserver ها از نوع cache-only هستند زیرا اطلاعات کاملی را درمورد نام ها و آدرس های IP ندارند و براساس request کاربران از روی

Name server های primary یا secondary آپدیت می شوند.

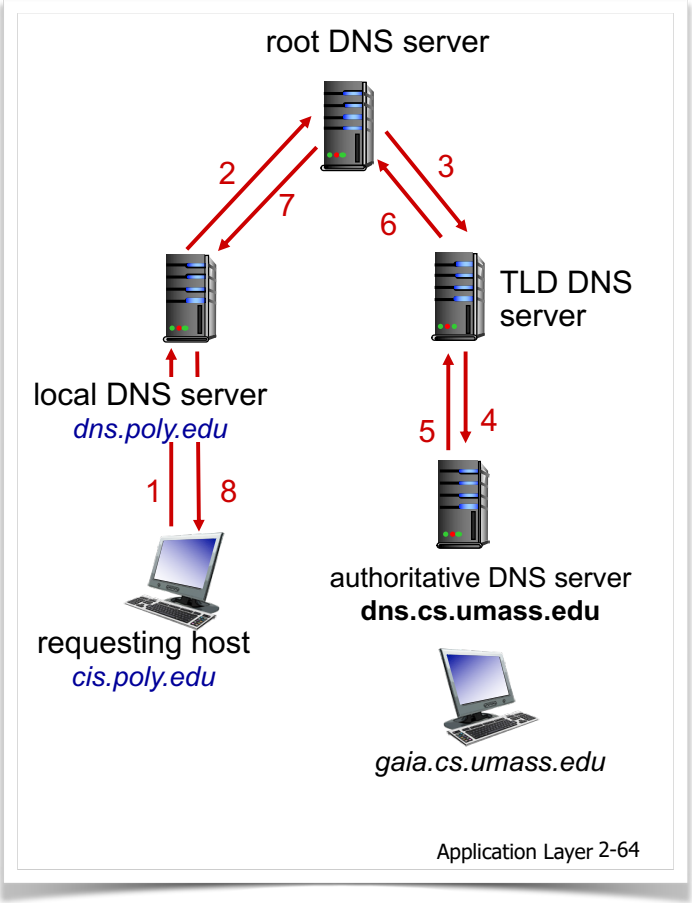
name server های local در واقع name server های cache-only هستند که در خواست های مختلف را به دیگر name server ها می فرستند.

ابتدا از name server، cache-only پرسیده می شود که آدرس متناظر با نام موردنظر را دارد یا خیر، اگر داشت پاسخ می دهد، اگر نداشت به root name server می گوید، سپس root name server به TLD name server ارجاع می دهد و آن هم به یک authoritative nameserver ارجاع می دهد و آن authoritative nameserver دیگر host موردنظر را می شناسد و به آن پاسخ می دهد.

iterative


root DNS server
2
3
TLD DNS server
4
5
local DNS server
dns.poly.edu
1
8
7
6
authoritative DNS server
dns.cs.umass.edu
requesting host
cis.poly.edu
gaia.cs.umass.edu
Application Layer 2-63

Recursive :



در روش Recursive، local DNS server به root DNS server خود در خواست می دهد و دیگر root DNS server مراحل بعدی را هندل می کند و این روش سربار دارد.

سه مورد از معروف ترین DNS Server ها عبارتند از :

Popular DNS Servers

Name	Address	Type
Cloudflare	1.1.1.1 and 1.0.0.1	Primary and Secondary
Google Public DNS	8.8.8.8 and 8.8.4.4	Primary and Secondary
Quad9	9.9.9.9 and 149.112.112.112	Primary and Secondary
OpenDNS	208.67.222.222 and 208.67.220.220	Primary and Secondary
Comodo Secure DNS	8.26.56.26 and 8.20.247.20	Primary and Secondary

۴) پورت پیشفرض مورد استفاده در پروتکل DNS چیست؟

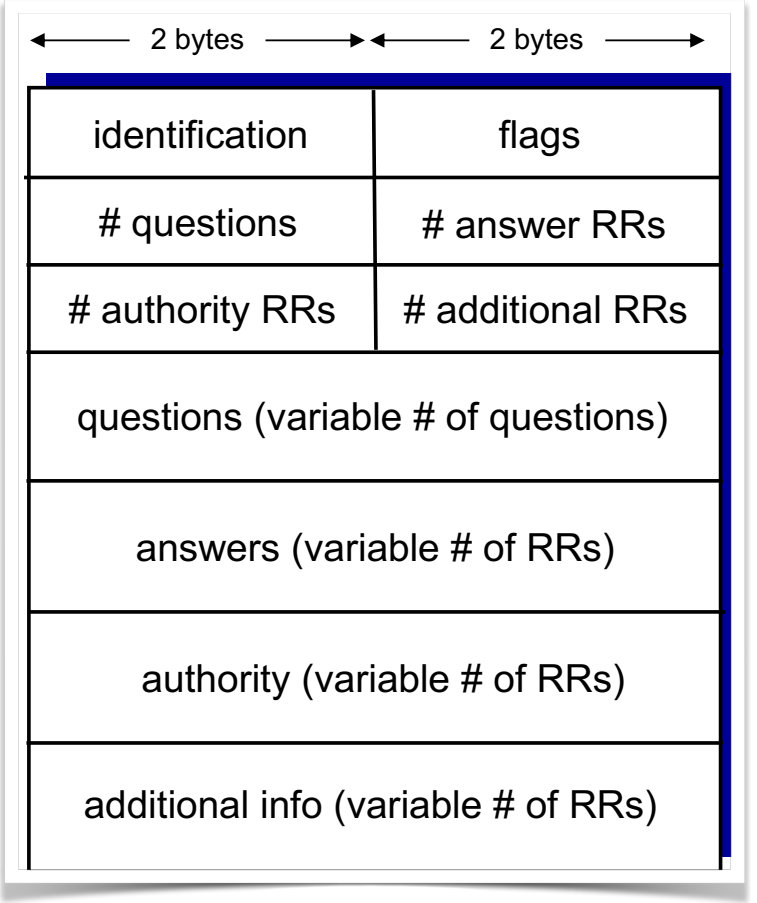
پورت ای که برای DNS بسیار مورد استفاده قرار می گیرد UDP 53 است، این زمانی استفاده می شود که یک client با یک DNS server ارتباط برقرار می کند و ماکزیمم سایز query مجاز در این حالت 512 بایت است.

اما TCP 53 زمانی استفاده می شود که می خواهیم zone transfer انجام دهیم، همچنین query بیشتر از 512 بایت است.

در سوال ۶ گفته ایم که در چه سناریوهایی از UDP و در چه سناریوهایی از TCP استفاده می کنیم.

در واقع زمانی از UDP استفاده می کنیم که داده های اصلی از کم هستند و سرعت برایمان خیلی مهم است و برای دلایل دیگر به جواب سوال ۶ مراجعه شود:)).

۵) ساختار بسته های DNS به چه شکل می باشد؟



ساختار بسته های DNS به شکل فوق است که به بررسی هر مورد از ساختار این بسته می پردازیم :

ارائه ی خدمات سرویس DNS بین یک resolver و name server در قالب مجموعه ای از پیغام ها انجام می شود زیرا message passing داریم، حال این پیغام هایی که می فرستیم یک فرم مشخص به شکل فوق دارند، و این فرم هم فرمت query

است و هم فرمت reply، اولین فیلد که دو بایت برایش درنظر گرفته شده است، فیلد identification یا هویت است، زمانی که ما query می فرستیم و در قالب query ها، response دریافت می کنیم، باید بین query هایی که می فرستیم و response هایمان ارتباط برقرار کنیم، identification یا هویت مشخص می کند که هر response مربوط به چه query ای است، ارتباط بین query و response را برای ما تعیین می کند.

فیلد بعدی که دو بایت است، flags یا پرچم ها است، در این 16 بیت تعیین می کنیم که این پیغام از نوع query است یا reply(یک بیت برای تعیین مورد استفاده قرار می گیرد، اگر این بیت را 0 درنظر بگیریم به این معناست که داریم query می فرستیم و درصورتی که آن را 1 درنظر بگیریم به معنای این است که داریم reply می فرستیم).

Flag بعدی در این 16 بیت Flag، authoritative است، آیا reply ای که در حال ارسال است از یک name server ای ارسال می شود که authoritative یا خیر اگر authoritative است این بیت را 1 می کنیم اگر name server، cache-only است این بیت را 0 می کنیم.

Flag بعدی در این 16 بیت، recursion desired است، زمانی که مقدار دهی می شود به این معنا است که فرستنده درخواست recursive name resolution را دارد، میخواهد که name server برای او به صورت recursive نام را تبدیل به آدرس IP کند، حال اگر امکان recursion موجود باشد، Flag بعدی که recursion available است را سرور به 1، set می کند.

در قسمت questions اطلاعات مربوط به query قرار می گیرد، نام مدنظر برای پرس جو، تایپ رکوردی که می خواهیم مورد پرس جو قرار بدهیم و در قسمت answers نیز RR های ما قرار می گیرند، سپس در قسمت authority# تعداد name server هایی که به عنوان authoritative در سطح آن دامنه هستند قرار می گیرد و اطلاعات مربوط به آن name server ها نیز در بخش authority قرار می گیرد.

توجه شود که questions# و answers# نیز تعداد سوالات و جواب ها و رفرنسی به questions و answers قرار داده می شود، در قسمت additional هم اطلاعات جانبی که در مورد یک query در request ما موجود است قرار می گیرد و تعداد اطلاعات اضافی که در body قرار داده ایم در additional# قرار می گیرد و اطلاعات اضافی را نیز در قسمت additional قرار می دهیم مثلا فرض کنیم رکورد تایپ A را مورد پرس جو قرار داده ایم چند آدرس IP داریم، یک آدرس IP به عنوان آدرس اصلی است و آدرس های اضافی در قسمت additional قرار دهیم یا چند پیغام جداگانه برای هر آدرس IP از سمت گیرنده ارسال شود.

فرمت خود RR ها نیز به این صورت است(برای اطلاعات بیشتر به پاسخ سوال 2 مراجعه شود) :

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

اما می خواهیم این ساختار را هر چه دقیق تر بررسی کنیم :

DNS Packet Structure

Header	
Question	Question for the name server
Answer	RRs answering for question
Authority	RRs pointing toward an authority
Additional	RRs holding additional information

Header

Header Structure

ID
QD | Opcode | AA | TC | RD | RA | Z | RCODE
QDCOUNT
ANCOUNT
ARCOUNT

این قسمت همیشه در وجود دارد و نشان می دهد که چه قسمت هایی از ساختار اصلی بسته مانند Answer، Additional و Authority وجود دارند.

ID :

یک عدد 16 بیتی است به اسم Identifier که Query زده شده و response مربوط به آن را به هم وصل می کند. (mapping)

QR :

یک بیت است و نشان می دهد که Query است یا response.

If Query then
QR = 0
If Response then
QR = 1

OPCODE :

یک عدد 4 بیتی است که نشان دهنده ی نوع Query است.
این مقدار توسط کسی که Query را می فرستد مقدار دهی می شود و در Response نیز کپی می شود.

OPCODE

OPCODE	Meaning
0	A standard query (Query)
1	an inverse query (IQUERY)
2	a server status request (STATUS)
3-15	reserved for future use

AA(Authoritative Answer) :

این بیت در response، valid است و مشخص می کند که name server ای که به ما پاسخ داده است authority است یا خیر.

TC :

نشان می دهد که آیا message کوتاه شده است بدلیل بزرگتر بودن طول آن نسبت به مقداری که برروی کانال ارتباطی مجاز به ارسال هستیم.

RD(Recursion Desired) :

این بیت برای انجام Query به صورت recursive ، set می شود.

RA(Recursion Available) :

این بیت در Response نشان می دهد که آیا recursive query ساپورت می شود یا خیر.

Z :

برای استفاده ی آینده رزرو شده است و باید مقدار 0 را داشته باشد در تمام Query ها و Response ها.

RCODE(Response Code) :

این یک عدد 4 بیتی است که به عنوان قسمتی از Response، set می شود.

RCODE

OPCODE	Meaning
0	No error condition

OPCODE	Meaning
1	Format error - The name server was unable to interpret the query.
2	Server failure - The name server was unable to process this query due to a problem with the name server.
3	Name Error - Meaningful only for responses from an authoritative name server, this code signifies that the domain name referenced in the query does not exist.
4	Not Implemented - The name server does not support the requested kind of query.
5	Refused - The name server refuses to perform the specified operation for policy reasons. For example, a name server may not wish to provide the information to the particular requester, or a name server may not wish to perform a particular operation (e.g., zone transfer) for particular data.
6-15	Reserved for future use.

QDCOUNT :

یک عدد بدون علامت 16 بیتی است برای نشان دادن تعداد Question entry ها.

ANCOUNT :

یک عدد بدون علامت 16 بیتی است برای نشان دادن تعداد Resource Record ها در قسمت Answer.

NSCOUNT :

یک عدد بدون علامت 16 بیتی است برای نشان دادن تعداد name server resource record ها در قسمت Authority.

ARCOUNT :

یک عدد بدون علامت 16 بیتی است برای نشان دادن تعداد Resource Record ها در قسمت Additional.

Question

Question Section Format

Structure	Description
QNAME	a domain name represented as a sequence of labels, where each label consists of a length octet followed by that number of octets. The domain name terminates with the zero length octet for the null label of the root. Note that this field may be an odd number of octets; no padding is used.

Structure	Description
QTYPE	a two octet code which specifies the type of the query. The values for this field include all codes valid for a TYPE field, together with some more general codes which can match more than one type of RR.
QCLASS	a two octet code that specifies the class of the query. For example, the QCLASS field is IN for the Internet.

Answer, Authority, Additional

این سه قسمت همگی داری یک ساختار هستند :

Resource Record Format

Resource Record Format

Structure	Description
NAME	a domain name to which this resource record pertains.
TYPE	two octets containing one of the RR type codes. This field specifies the meaning of the data in the RDATA field.
CLASS	two octets which specify the class of the data in the RDATA field.

Structure	Description
TTL	a 32 bit unsigned integer that specifies the time interval (in seconds) that the resource record may be cached before it should be discarded. Zero values are interpreted to mean that the RR can only be used for the transaction in progress, and should not be cached.
RDLENGTH	an unsigned 16 bit integer that specifies the length in octets of the RDATA field.
RDATA	a variable length string of octets that describes the resource. The format of this information varies according to the TYPE and CLASS of the resource record. For example, the if the TYPE is A and the CLASS is IN, the RDATA field is a 4 octet ARPA Internet address.

توجه شود که برخی از موارد را انگلیسی بیان کردیم تا زیرا متن انگلیسی آن باعث درک بهتر مطلب می شود.

۶) دلیل توصیه RFC برای استفاده از پروتکل UDP در Queryها نسبت به TCP چیست؟

ابتدا نیاز است یک سری توضیحات اولیه ارائه دهیم :

برنامه های کاربردی یک سری نیازمندی ها دارند که این نیازمندی ها را از زیرساخت شبکه از لایه ی Transport دریافت می کنند، در لایه ی Transport، Application قرار دارد که سرویس های لایه ی Transport را ارائه می دهد و در سطح لایه ی Application، برنامه نویس قرار دارد و برنامه های خود را توسعه می دهد و برای اینکه بین برنامه نویس لایه ی Application و Application لایه ی Transport ارتباط برقرار کنیم به API نیازمندیم(یعنی در توسعه ی نرم افزار های کاربردی درگیر پیچیدگی های زیرساخت شبکه نمی شویم و به پیاده سازی هایی که در آن سطح انجام شده اند کاری نداریم)

آن چیزهایی که در لایه ی Application است مانند اینکه syntax یا semantics هر پیغام چیست و ... قوانینی که براساس آن می توان فهمید چه زمانی ارسال و دریافت پیغام انجام دهیم اینها به عهده ی ما است اما در سطح لایه ی Transport پارامترهایی که انتقال داده را ممکن می کنند همه در اختیار ما نیست اما می توانیم یک سری تنظیمات کلی را انجام دهیم، مثلا اینکه پروتکلی که استفاده می کنیم چیست؟ UDP است یا TCP؟

حال باید ببینیم بسته به نوع Application چه سرویس لایه ی Transport ای را باید مورد استفاده قرار دهیم.

Reliable transport :

سرویس TCP امکان انتقال اطلاعات به صورت reliable را برای ما فراهم می کند، یعنی تضمین می کند که داده ها به

مقصد می رسند و به همان ترتیبی که ارسال می شوند در مقصد دریافت می شوند در مقابل UDP این امکان را ندارد و اگر داده ای در طول مسیر دچار خطا شد TCP آن را مجددا ارسال می کند و برای هر داده ای هم که در مقصد دریافت می شود یک تاییدیه مبنی بر دریافت آن خواهد از مقصد خواهد گرفت.

اما اگر در UDP به reliablity نیاز پیدا کنیم باید آن را در سطح لایه ی Application بنویسیم.

Flow of control :

یک زمانی کامپیوتر فرستنده، کامپیوتر قوی است و قدرت پردازشی بالایی دارد و با یک نرخ بالایی شروع به تولید داده می کند و گیرنده کند است یا دچار ازدحام است و request های زیادی برای آن آمده است و نمی تواند با نرخ ارسال فرستنده پردازش را انجام دهد، در این حالت گیرنده به فرستنده اعلام می کند که نرخ انتقال خود را پایین تر ببرد.

در TCP این مسئله هندل میشود اما در UDP اینطور نیست.

Congestion control :

فرستنده و گیرنده خوب عمل می کنند اما شبکه دچار ازحام است و نمی تواند داده را خوب منتقل کند، در این حالت نیز به فرستنده اعلام می کند که نرخ انتقال را پایین بیاورد و با سرعت کمتری داده بفرستد و این را TCP هندل می کند اما در UDP اینطور نیست.

هردو پروتکل هیچ تضمینی در این باره که اگر داده ای اولویت دارد از لحاظ زمانی آن را زودتر بفرستد.

هردو پروتکل هیچ تضمینی در حداقل پهنای باند نداریم.

هردو پروتکل اینطور فرض کرده اند که کسانی که از شبکه استفاده می کنند trusted هستند و امنیت را هندل نمی کنند.

در TCP برای مدل ارتباطات خودش از مدل connection-oriented استفاده می کند، یعنی connection می سازد و بعد داده میفرستد اما در UDP اینطور نیست.

در واقع این connection به صورت مجازی و نرم افزاری است صورت می گیرد و بخشی از پهنای باند یا زمان در اختیار connection نمی گیرند و به صورت فیزیکی و سخت افزاری نیست، یعنی منابع شبکه کنده نمی شوند و صرفا برای دریافت تضمینی داده به صورت صحیح در مقصد است که این کار را نیز UDP انجام نمی دهد.

اما به چه دلیل در برخی از سناریو ها UDP به TCP ترجیح داده می شود با اینکه TCP مزیت های بیشتری دارد؟

دلیل آن بالا بودن سرعت و نرخ انتقال بالای UDP است و دلیل بالا بودن سرعت آن overhead کم آن است، در واقع TCP برای ارائه سرویس هایی که گفتیم باید داده های کنترلی به بسته اضافه کند و این اطلاعات کنترلی در واقع header می شوند و این یعنی هدر دادن پهنای باند یعنی در TCP داده ی کمتری را می فرستیم و اطلاعات کنترلی بیشتری را در بسته قرار می دهیم اما در UDP اطلاعات کنترلی کم است زیرا سرویس های UDP کم است و در نتیجه داده ی بیشتری نیز منتقل می کنیم و سرعت UDP بیشتر است.

یک زمانی ما قابلیت اطمینان نمی خواهیم در نتیجه از UDP استفاده می کنیم.

گاهی اوقات داده هایی که می فرستیم آن قدر کم هستند که اگر در بین راه خراب شوند Application دوباره آن را می

فرستد یعنی اون هزینه ی overhead، TCP خیلی بیشتر از انتقال مجدد دیتا است پس از UDP استفاده می کنیم مثلا مانند TFTP(مثلا در تجهیزات شبکه استفاده می شود) که برای انتقال فایل های کوچک است اما FTP که برای انتقال فایل های حجیم است از TCP استفاده می کند.

زمانی هم وجود دارد که ماهیت سرویس در لایه ی Application، بصورت request response است ، مانند سرویس های پایگاه داده، یعنی زمانی که query می گیریم یک response میاد و زمانی که این query ایراد داشته باشد response درستی هم نخواهیم داشت یعنی خود Application این را می فهمد و درک می کند که داده ی ارسالی اش دچار خطا شده است یا خیر پس دیگر نیازی نیست که این سرویس را در لایه ی Transport پیاده سازی اش کنیم یعنی duplication رخ می دهد، پس درنتیجه از UDP استفاده می کنیم.

۷) سوکت چیست؟

سوکت یک API است (socket API).

اما خود API به چه معناست؟

API مخفف Application Programmer Interface است و در واقع یک واسط بین برنامه ی کاربردی و برنامه نویس است، برنامه نویس در لایه ی Application قرار می گیرد، برنامه نویسان در لایه ی Application برنامه های کاربردی خود را توسعه می دهند، حال Application می تواند سرویس های لایه ی Transport در سطح سیستم عامل باشد، که ما با استفاده از سرویس های آن می توانیم به زیر ساخت شبکه متصل شویم. سوکت در واقع یک API است بین برنامه ی کاربردی که ما توسعه داده ایم و خدمات لایه ی Transport برای دستیابی به زیرساخت شبکه.

برنامه های کاربردی تحت شبکه برای اینکه بتوانند از زیرساخت شبکه برای تبادل اطلاعات استفاده کنند از سوکت استفاده می کنند.

در واقع سوکت مانند دری است که با استفاده از آن پیغام هایمان را از طریق آن منتقل می کنیم.

زمانی که می خواهیم از مفهموم سوکت استفاده کنیم در سطح لایه ی Application برنامه نویس کنترل کامل دارد، اما در سطح لایه ی Transport سوکت تمام امکانات را در اختیار کاربر قرار نمی دهد یک سری موارد مانند تعیین نوع پروتکل لایه ی Transport مثلا TCP است یا UDP، تعیین بعضی از پارامتر ها مانند اندازه segment لایه ی Transport یا اندازه ی بافر اینها بوسیله ی برنامه نویس قابل کنترل هستند اما بقیه

ی موارد بوسیله ی سیستم عامل کنترل می شود و کنترل آن از دست برنامه نویس خارج است.

مثلا می گوییم من میخواهم یک سوکت از نوع TCP ایجاد کنم با یک سرور با آدرسIP مشخص و Port مشخص و اتصال را برقرا کنم، تمام موارد بصورت API در سوکت تعریف شده اند و ما از interface آن استفاده می کنیم و به جزییات پیاده سازی آن کنترلی نداریم و سیستم عامل است که کارها را در بستر شبکه هندل می کند.

ترکیب یک آدرس و پورت باهم مثلا :

192.168.2.1:80

یک سوکت را برای ما مشخص می کند(ترکیب آدرس HOST و پورت آن).

THE END