The Evolution of the DNS

Geoff Huston AM
Chief Scientist, APNIC

Why…

Are we interested in the Internet's name infrastructure, given that APNIC is an IP address registry operator?

- Because names and addresses are dependant on each other to provide a common and coherent infrastructure for Internet applications
- Given the fractures in the address infrastructure because of the incomplete deployment of IPv6 and the extensive use of NATs, we are increasingly relying on the name infrastructure to provide coherence to the Internet
- So our interest in Internet infrastructure extends to names and namespaces and the DNS as a name resolution protocol

So lets dive into the world of names and the DNS!

The Origins of the DNS

- The DNS was created as a replacement for a static list of hosts and addresses - /etc/hosts
  - Which was a list of host names and their IP addresses

```
localhost           127.0.0.1
example.com         192.0.2.1
```

  - To resolve a name you look up the name in /etc/hosts.txt and use the result

Question: What are the problems with this approach?

Next Steps

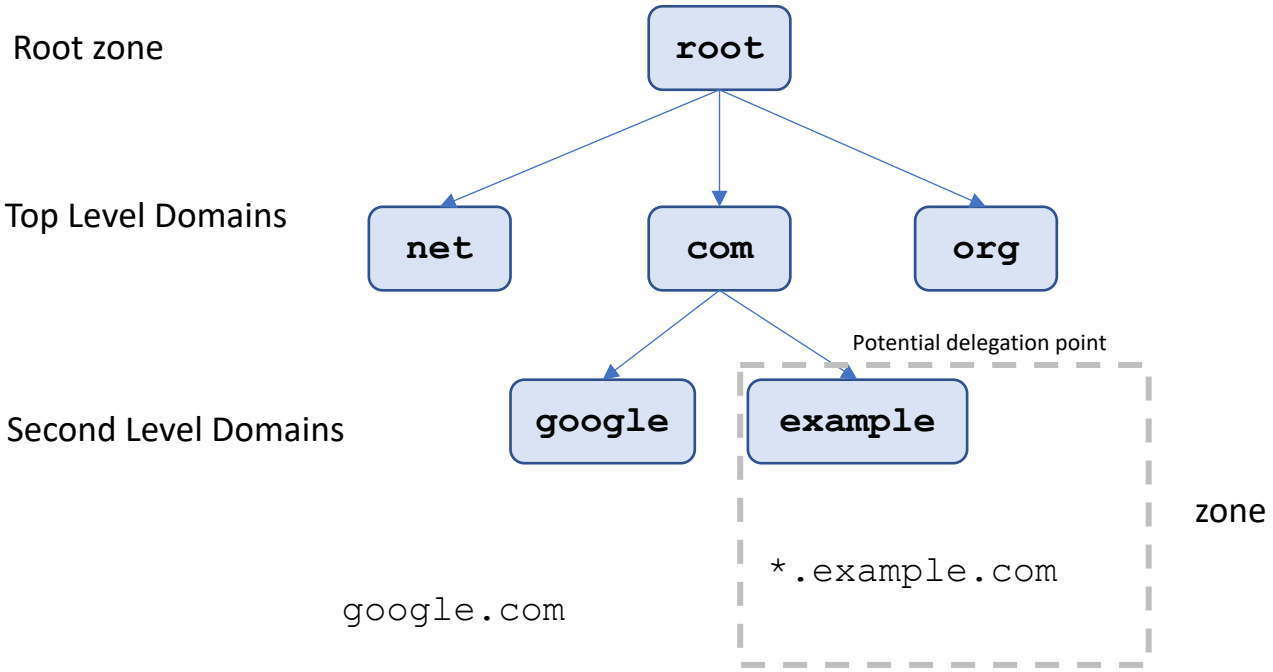
- Transform local name lookup into a network service
  - That way we can use a local *nameserver* to serve the local copy of the hosts file to a collection of local clients
  - Better scalability, as the updates need only to be sent to each network's local name server, and network clients simply query this server for name-to-address translation

- IEN 61 – October 78
  - Simple datagram exchange: send the name server a packet with a query name and receive a response with the original query and the IP address added
  - Allows one host to serve its copy of the hosts file with a collection of clients

Next Steps

RFC 822 – November 1983

- “tree-structured” name hierarchy
- Multiple “types” can be associated with each label
- Defines aliases (CNAME) and wildcards
- Distributed set of name servers aligned with the distributed name structure
- Resolvers to traverse the name server structure to resolve a name
- DNS protocol defined as a simple query/response datagram protocol



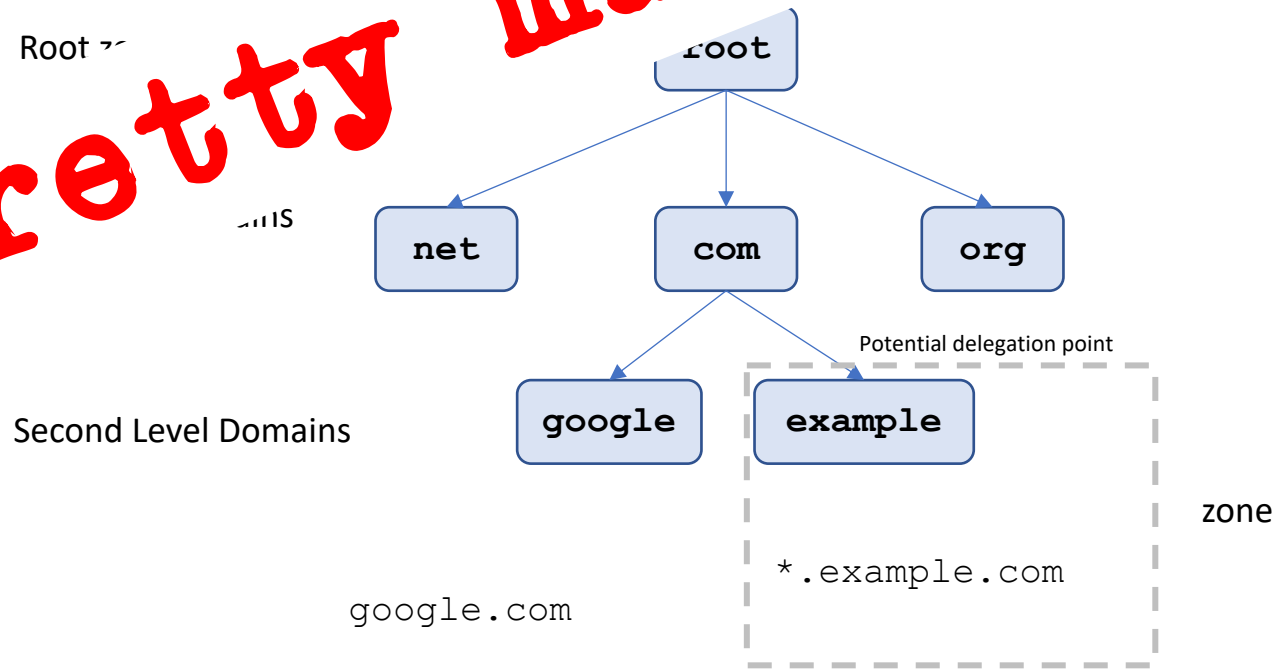
Next Steps

RFC 822 – November 1983

- "tree-structured" name hierarchy
- Multiple "types" can be associated with eac label
- Defines aliases (CNAME) and
- Distributed set of na the distribute
- Reso server

a simple query/response protocol

And that's pretty much it!

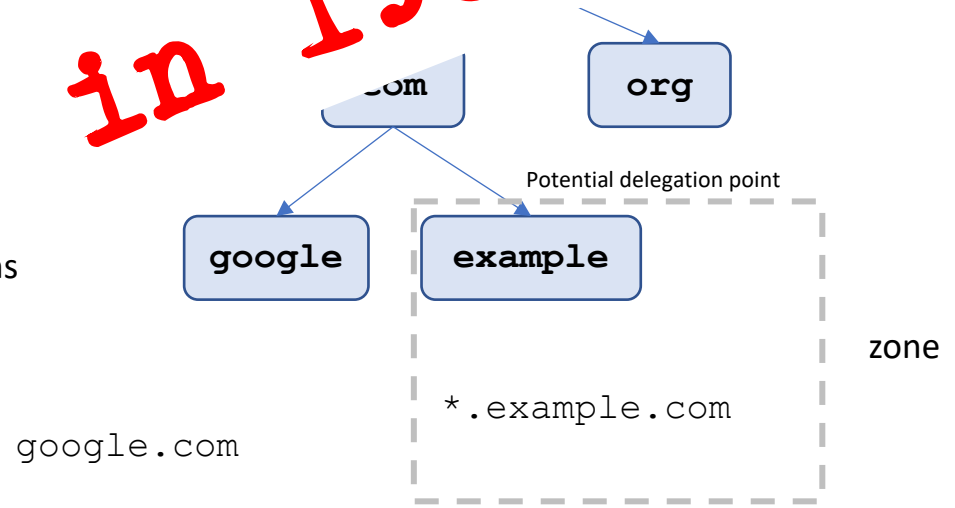


Next Steps

RFC 822 – November 1983

- "tree-structured" nam
- Multiple "type
  each lab
- R
- D
  da

Everything since then has just been cleaning up fine details – the DNS today is largely as it was in 1983!



End.

OK, maybe there is more to the DNS story

The DNS is not perfect:

- It can be extremely slow!
- It leaks information like crazy
- It's prone to manipulation and disruption
- It's rigid
- It's insecure
- It's a source of incredibly effective DOS attacks

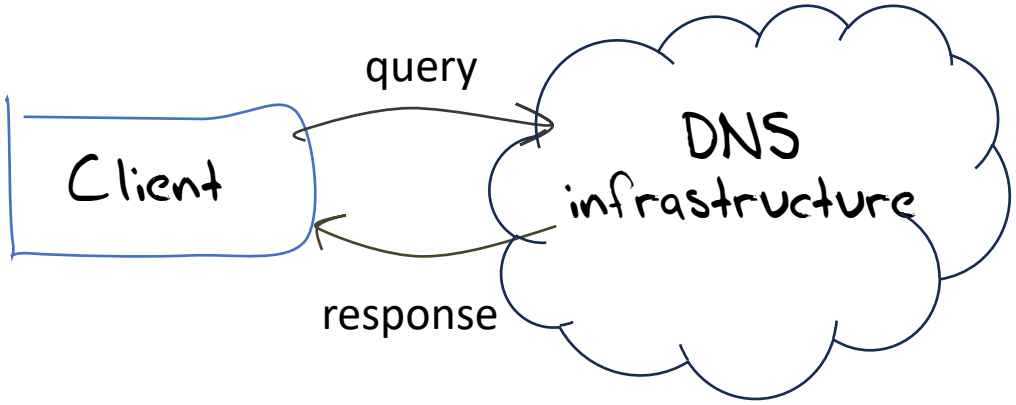
- For a common service that everybody uses its not exactly a paragon of good engineering design
- So there are continual efforts to make the DNS "better"

What are we doing about making the DNS "better"?

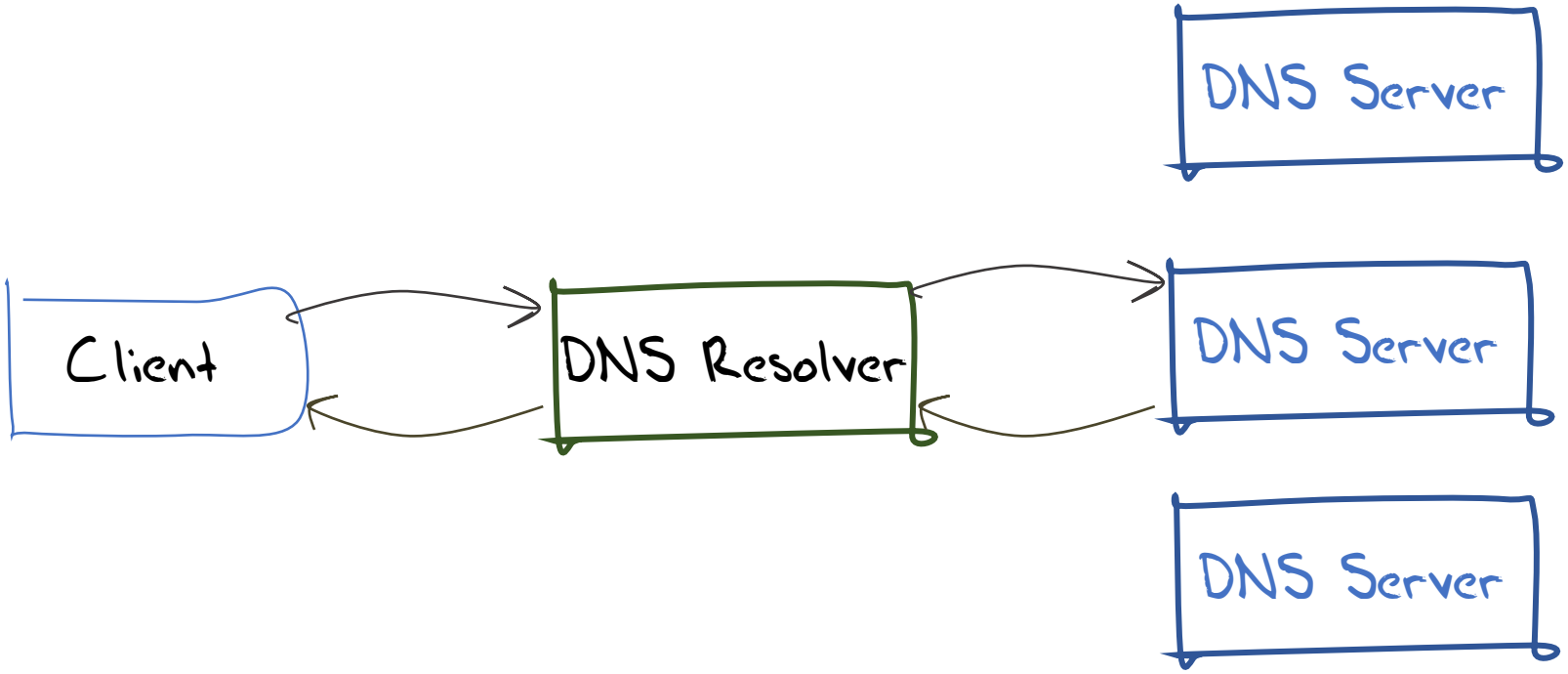
Aside: The DNS is insanely complex!

- For a simple distributed database structure and an equally simple query/response resolution protocol, the DNS has ended up being both complicated and complex in all kind of ways
- Which makes evolution of the DNS “tricky”
- What is “the DNS”?

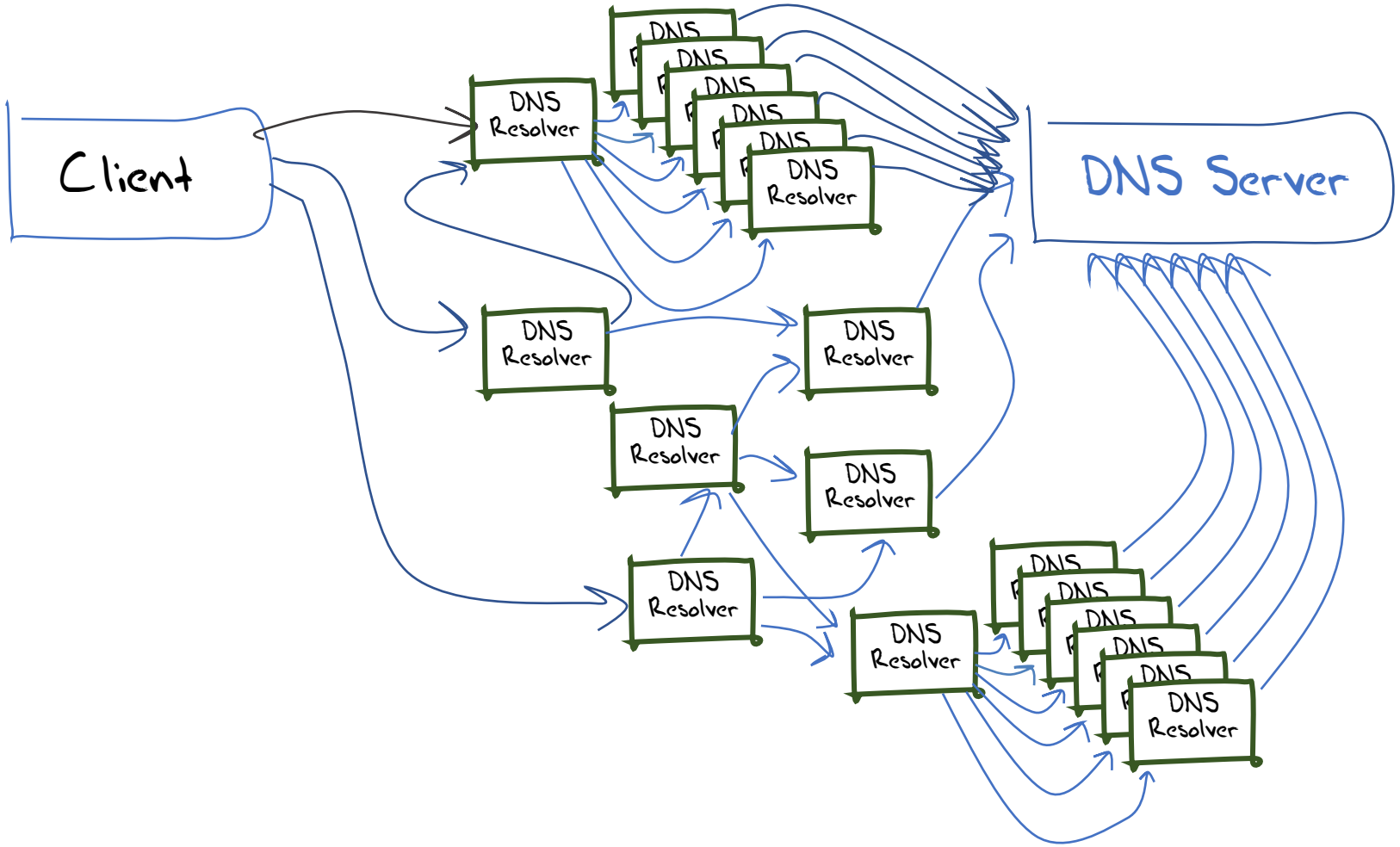
A DNS model



How we might like to think the DNS works



What we suspect the DNS is like



The DNS is a mystery

- Noone can track where your query might go
- Noone can say how many additional queries you might trigger
- Noone can tell where your answer came from
- Its really hard to tell if the answer is correct

DNS Privacy Issues

- Lots of actors get to see what I do in the DNS
  - My platform
  - My ISP's recursive resolver
  - Their forwarding resolver, if they have one
  - Authoritative Name servers
  - Snoopers on the wire
- Can we make it harder for these "others" to snoop on me?

I. DNS Privacy - Qname Minimisation

- DNS name resolution has two parts: discovery of the name server of the terminal DNS zone, and then resolution by asking that name server the query name and query type
- The DNS “overshares” information using the full query name during discovery

DNS oversharing

The DNS uses the full query name to discover the identity of the name servers for the query name

Hi root server, I want to resolve www.example.com

Not me – try asking the servers for .com

Hi .com server, I want to resolve www.example.com

Not me – try asking the servers for example.com

Hi example.com server, I want to resolve www.example.com

Sure – its 93.184.216.34

The DNS is overly chatty

Is there an alternative approach to name server discovery that strips the query name in iterative search for a zone's servers?

- Yes – the extra information was inserted into the query to make the protocol simpler and slightly more efficient in some cases
- But we can alter query behaviour to only expose as much as is necessary to the folk who need to know in order to answer the query

Example of QNAME Minimisation

Ask the authoritative server for a zone for the NS records of the next zone:

Hi Root server, I want to know the nameservers for com

Sure, here are the servers for .com

Hi .com server, I want to know the nameservers for example.com

Sure, here are the servers for example.com

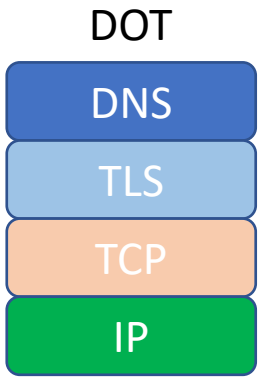
Hi example.com server, I want to resolve www.example.com

Sure – its 93.184.216.34

II. DNS Privacy - Channel Encryption

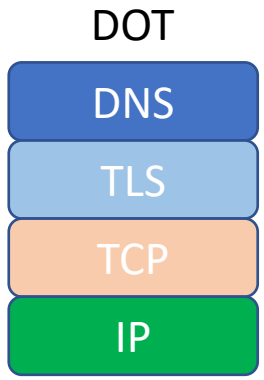
- Can we seal up DNS queries and responses such that they are not directly visible to any third party onlooker?

DNS over TLS (DOT)



- Similar to DNS over TCP:
  - Open a TLS session with a recursive resolver
  - Pass the DNS query using DNS wireline format
  - Wait for the response
- Can use held DNS sessions to allow the TLS session to be used for multiple DNS queries
- The queries and the responses are hidden from intermediaries
- Bonus: The client validates the recursive resolver's identity!

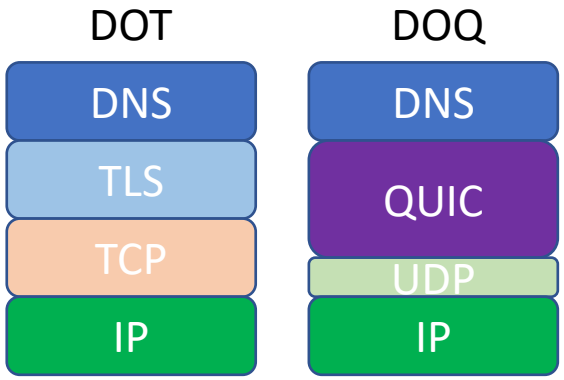
DNS over TLS (DOT)



Potential downsides to DoT:

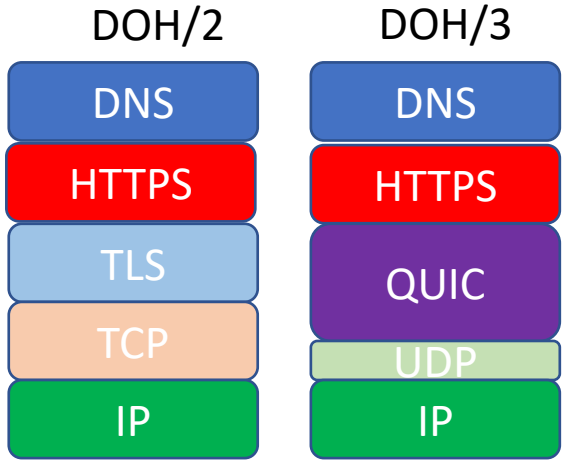
- Will generate a higher recursive resolver memory load as each client may have a held state with one or more recursive resolvers
- The TCP session state is on port 853
  - DNS over TLS can be readily blocked by middleware
- The privacy is relative, as the recursive resolver still knows all your DNS queries

DNS over QUIC (DoQ)



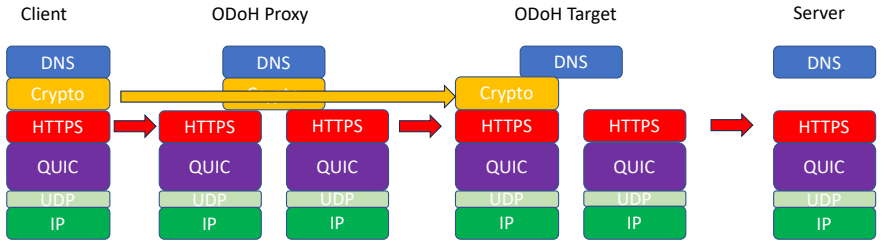
- QUIC is a transport protocol originally developed by Google and passed over to the IETF for standardised profile development
- QUIC uses a thin UDP shim and an encrypted payload
  - The payload is divided into a TCP-like transport header and a payload
- QUIC removes 1 RTT from session startup
- QUIC allows for interleaved queries without HOL blocking

DNS over HTTPS (DoH)



- DNS over HTTPS
- Uses an HTTPS session with a resolver
- Similar to DNS over TLS, but with HTTP object semantics
- Uses TCP port 443, so can be masked within other HTTPS traffic
- Uses DNS wireformat
- If the client and server support QUIC then the session will use HTTP/3

Oblivious DNS (oDNS)



- In all the previous approaches the recursive resolver knows the identity of the client and the client's DNS query
- Can we obscure this so that no one except the client itself knows both pieces of information?
  - Yes, Oblivious DoH
- oDNS uses a double encryption wrapper and two intermediaries:
  - An *ODoH proxy* which takes the user's encrypted query and send it onward using its own identity
  - An *ODoH target* which unwraps the query and passes it into the DNS
- Used in Apple's Private Relay product

Limitation of DNS Privacy

- None of these measures can assure you that the answer you get is authentic or not
- It just limits the number of “others” who might get to eavesdrop on your DNS queries and responses
- Detecting (and rejecting) tampering and manipulation within the DNS is a separate problem…

III. DNS Interference and Manipulation

- In many parts of the Internet the DNS is used to implement censorship provisions by interfering in the name resolution process
- The DNS also used in criminal attacks by redirecting potential victims to fake services
- How can a client distinguish between a genuine DNS response and a contrived lie?
  - DNSSEC!

What is DNSSEC?

(This answer could be really long or very short – I'll go for the ultra short version here)

- A DNS zone administrator generates a public/private key pair and then generates a digital signature for every authoritative record in the zone. These signatures are placed into the zone as RRSIG records. DNSSEC also signs across the "spans" between each pair of adjacent names in the zone. The public key is also placed into the zone as a signed record
- A hash of the zone's public key is passed to the zone's parent, which is placed into the signed parent zone as an authoritative (signed) DS record
- Clients can authenticate a DNS record by validating it against the associated signature record and assembling a validation chain from child to parent up to the root zone to validate the sequence of interlocking zone signing keys

DNSSEC Limitations

- DNSSEC can't tell a client what the "right" response might be. It can tell a client that the response that they have been provided is NOT authentic
- Validating signed responses entails assembling an interlocking signature chain from the target zone to the root
  - This will take some additional time to assemble this additional information from the DNS
- DNSSEC is also prone to generating large responses, so DNS over UDP can be a problem here

What is DNSSEC protecting you against?

- What's the threat issue going on here?
  - Kaminsky styled attack of off-path cache poisoning?
    - Between port and case randomisation there is probably adequate randomisation to protect the client from an off-path guessing attack
  - On path direct attack of response substitution?

- Even then - so what?
  - If we are looking at poisoning the name-to-address relationship and misdirecting the user then this is much the same as a routing attack -- TLS is going to help here by **authenticating the named identity** of the remote service – its IP address is irrelevant to this authentication!
  - If the service does not use some form of authentication then the client is very exposed in any case and DNSSEC is not going to mitigate all risks here!

DNSSEC is a Work in Progress

- For these reasons DNSSEC is still an active area of activity for the DNS
  - Can we make generating signatures “easier”?
  - Can we make validation faster?
  - What’s the use case to justify the incremental effort to sign and validation?
- It’s not clear if DNSSEC has an assured future
  - The added complexity and costs are very high compared to the quantification of incremental benefit

IV. Alternate Models

- The open nature of the DNS architecture has prompted many to innovate with the DNS in various “interesting” ways:
  - Why is there only one root zone? Can we set up a parallel DNS with alternate root zones?
  - Why are there exactly 13 root zone server names?
    - Can we experiment with a root zone that has a much larger set of named root servers?
    - Can we experiment with a root zone that has a single root server name?
  - What if we replace the hierarchy with a flat namespace space using distributed hash tables for name server discovery?
    - If we go in this direction, then can we make the system self-describing using blockchain technologies?
    - Can we combine keys, encryption and blockchains into a single secure and obscured name framework?

Blockchain DNS

- What if we replace the hierarchy with a flat namespace space using distributed hash tables for name server discovery?
  - If we go in this direction, then can we make the system self-managed using blockchain technologies to implement the common ledger?
  - Can we combine keys, encryption and blockchains into a single secure and obscured name framework?
    - GNU, Unstoppable Names, Etherum, Namecoin, Blockstack, Emercoin, +++
    - Some of these systems require modifgied clients to access the name space
    - Some use a “bridge name server” to translate the original DNS query into the customized namaespace for resolution

Dynamic DNS

- The DNS is a close to ubiquitously accessible service
- So you can think of DNS transactions as a form of remote procedure call:
  - Client invokes a server process with arguments defined in the query name
  - Server uses the server process outcomes to generate a response
  - This can be used for lightweight transactions, VPNs and IP tunnelling,

Where does this leave the DNS?

Evolution or Fragmentation?

- How should we think about these alternate name systems?
  - Are they experiments that attempt to innovate from the current model to address specific needs or shortcomings in the current model?
  - If such experiments gather momentum then we might expect that they would gradually be folded into the overall DNS framework
  - Or are these deliberate efforts to disrupt the current name system in an effort to divert revenue to the disruptor?

Evolution or Fragmentation?

- Should we be concerned?
  - Hard to say!
  - There a larger debate about coherence and stasis, as compared to innovation, flexibility and adaptation

  - If efforts to change the current framework succeed we call it “innovation”
  - If they fail, we call it “fragmentation”

Thanks!