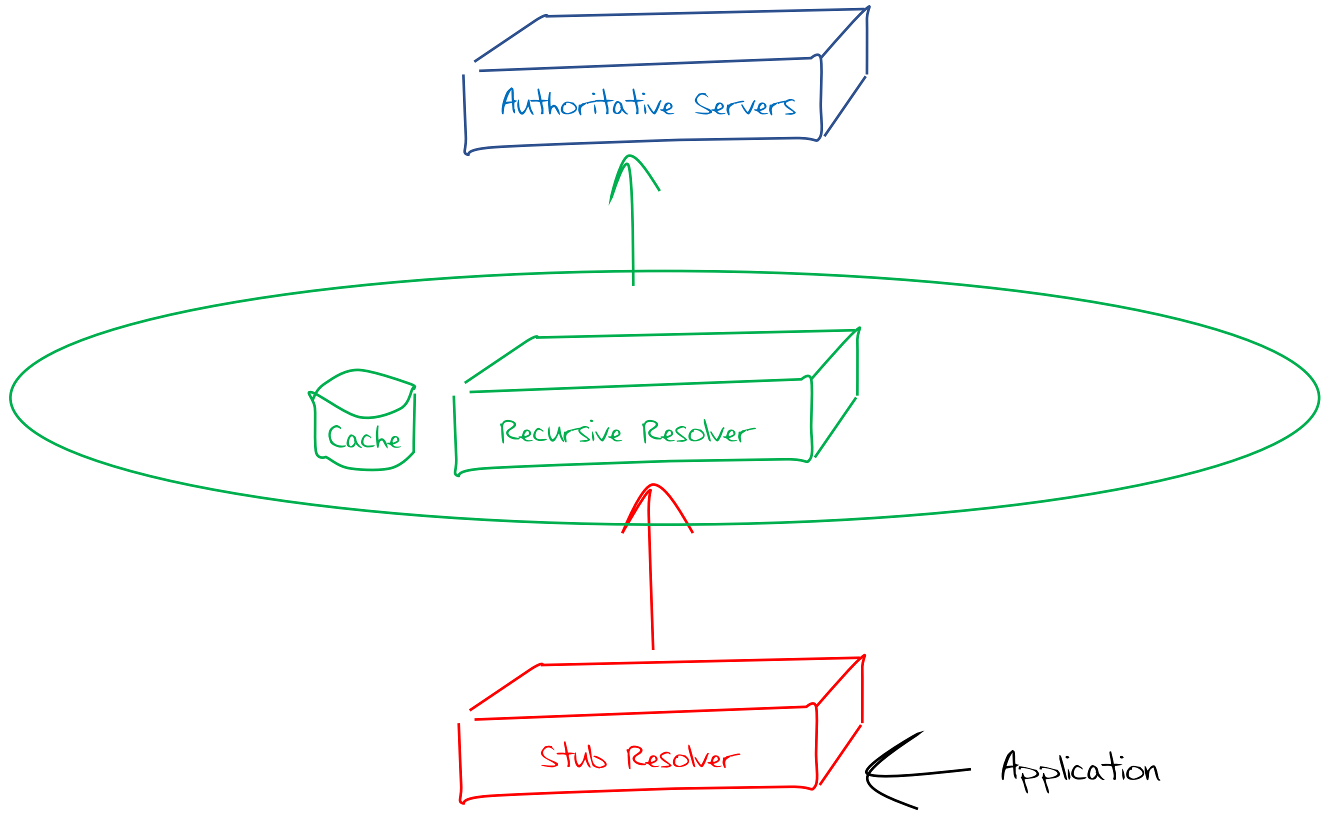
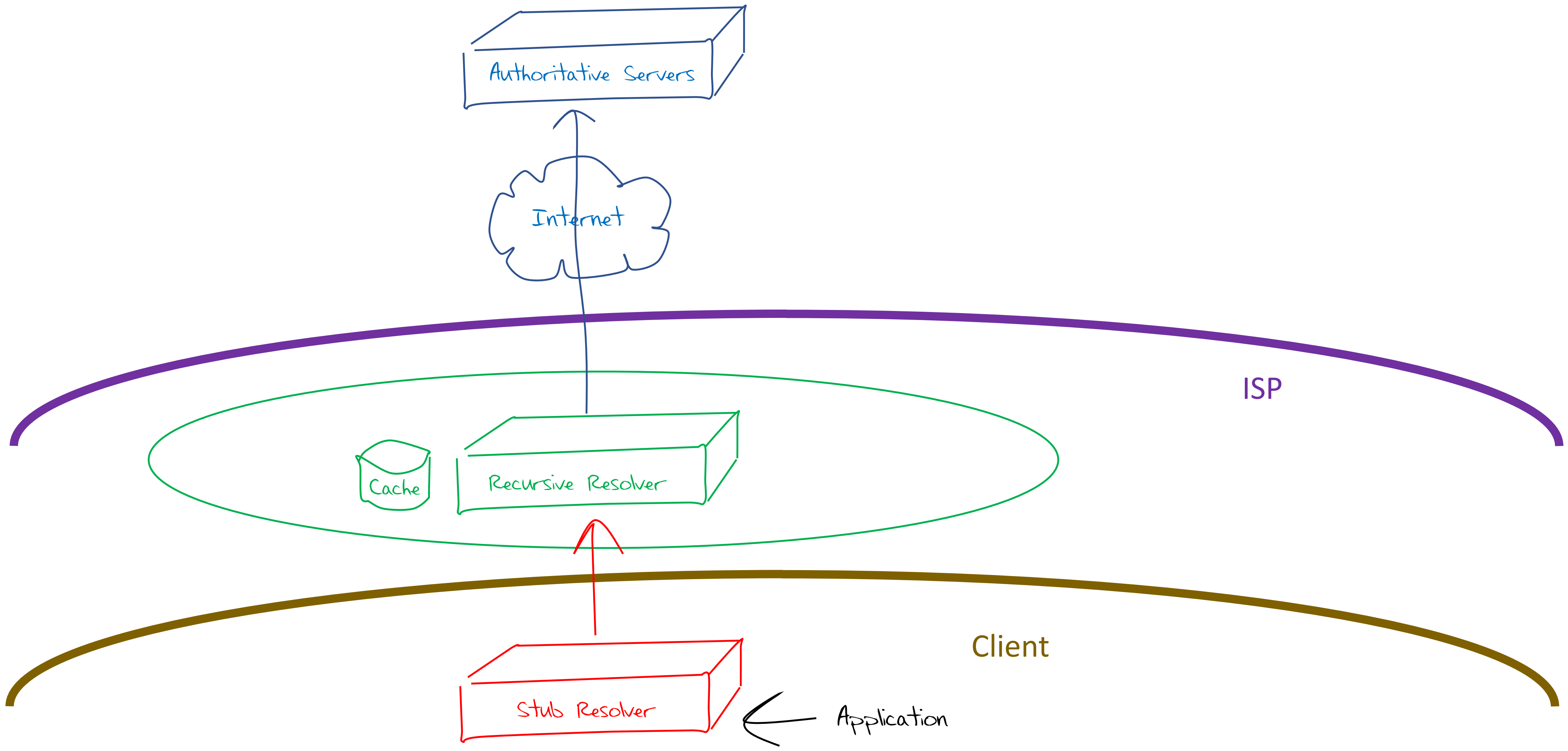
The Path to Resolverless DNS

Geoff Huston AM
Chief Scientist, APNIC

DNS System Architecture

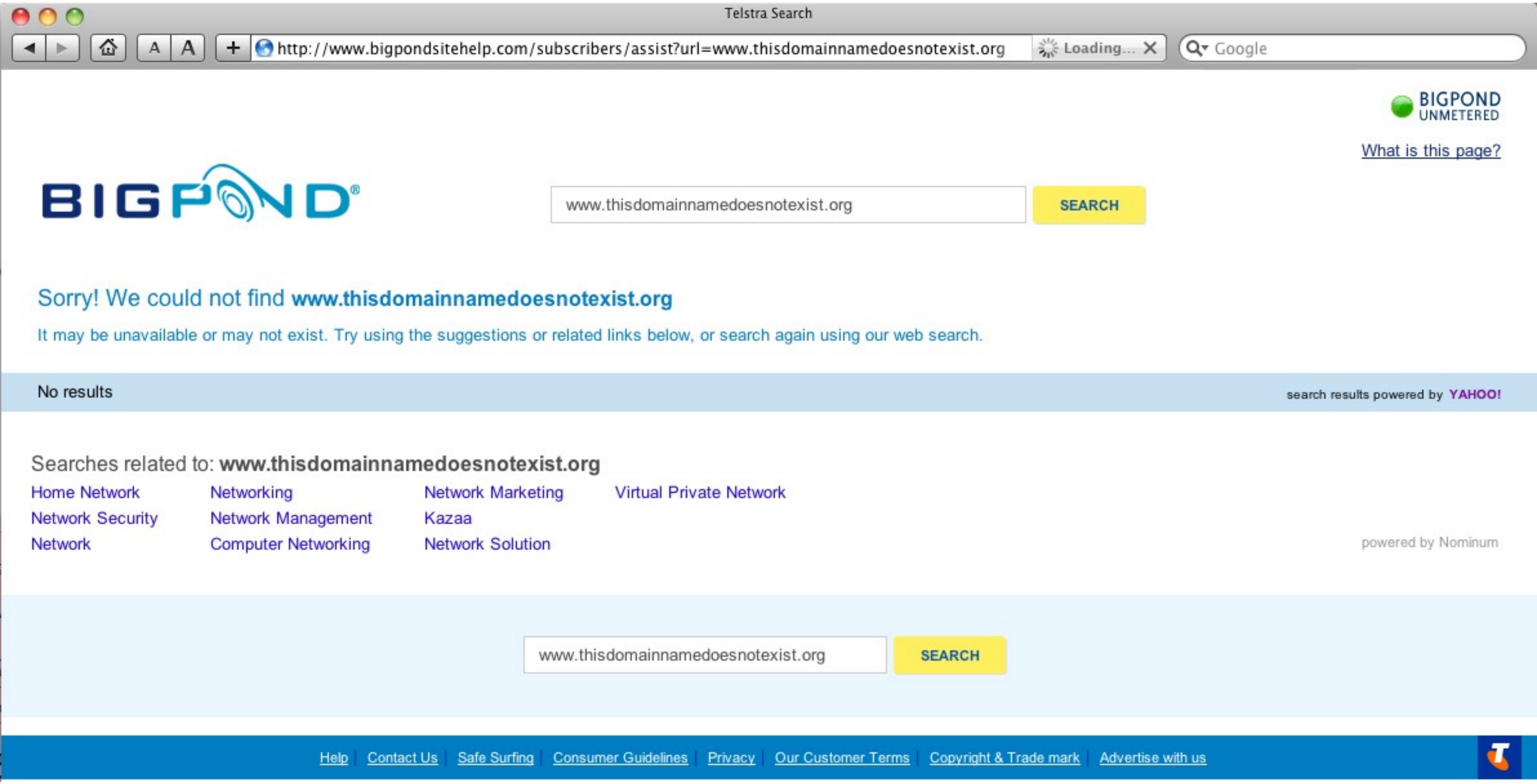


DNS System Architecture



Issues

- DNS Speed – the DNS can be exceptionally slow, and the interaction with resolver caches makes resolution unpredictable
- DNS Filtering – the DNS is a convenient control point for content management
- DNS MetaData collection – the DNS is a real time window on user behaviour
  - This can be performed at the recursive resolver, or by a third party on the path between the stub resolver and the recursive resolver
- DNS as Search – NXDOMAIN rewriting


Telstra Search
http://www.bigpondsitehelp.com/subscribers/assist?url=www.thisdomainnamedoesnotexist.org
Loading...
Google
BIGPOND UNMETERED
BIGPOND
SEARCH

What is this page?

www.thisdomainnamedoesnotexist.org

Sorry! We could not find **www.thisdomainnamedoesnotexist.org**

It may be unavailable or may not exist. Try using the suggestions or related links below, or search again using our web search.

No results

search results powered by YAHOO!

Searches related to: **www.thisdomainnamedoesnotexist.org**

Home Network	Networking	Network Marketing	Virtual Private Network
Network Security	Network Management	Kazaa	
Network	Computer Networking	Network Solution	

powered by Nominum

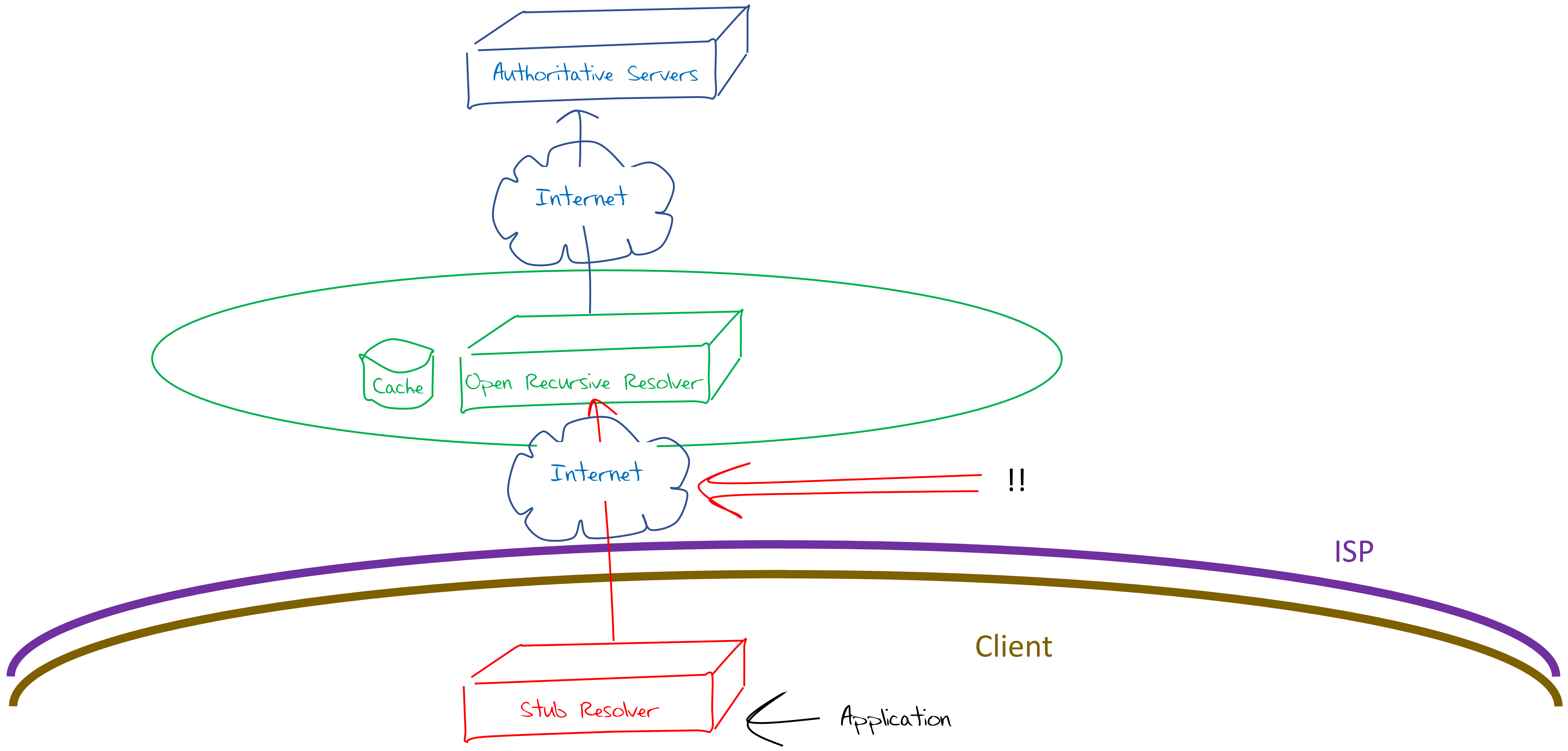
www.thisdomainnamedoesnotexist.org

Help | Contact Us | Safe Surfing | Consumer Guidelines | Privacy | Our Customer Terms | Copyright & Trade mark | Advertise with us

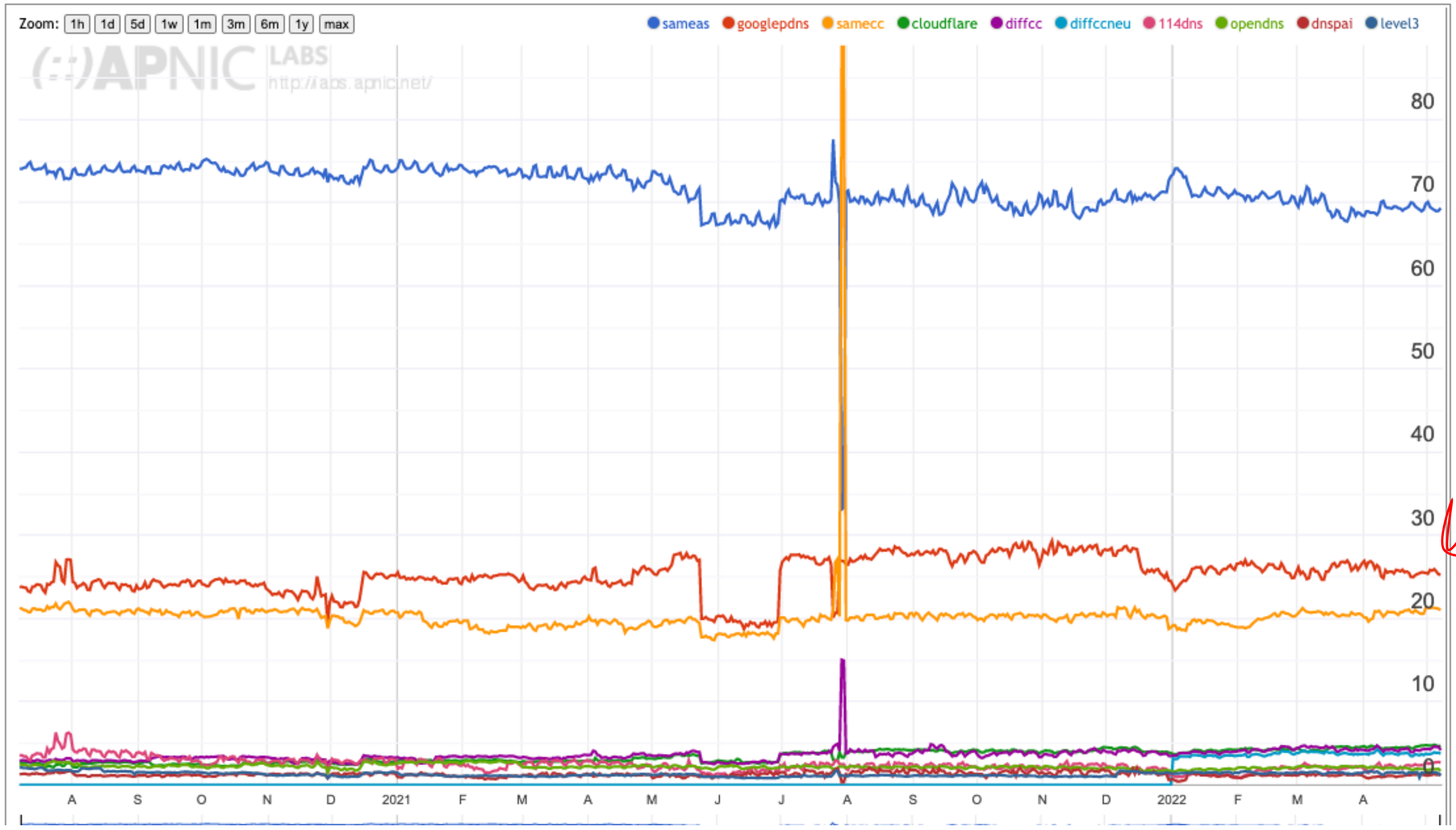
Background

- Browser vendors were simplifying the UI and combined the navigation and search input boxes to a single window
- This resulted in a significant level of cross leakage between DNS and search
- Some DNS resolver operators saw an opportunity to gather revenue by re-directing NXDOMAIN to search
- Google responded quickly with a large scale open DNS service that provided absolute integrity of responses, and supported DNSSEC validation to back this up
- Google’s DNS service rapidly gathered momentum, particularly in the enterprise sector

DNS System Architecture



Use of Resolvers in the Internet



68% of users direct their DNS queries to the ISP-operated recursive resolvers

25% of users direct their DNS queries to Google's public DNS service

20% of users direct their DNS queries in in-country DNS

https://stats.labs.apnic.net/rvrs

But…

- DNS queries use unencrypted UDP
  - Which itself is a huge DOS issue
- Combining this with a transit across the public Internet between the stub and open recursive resolver exposes further issues:
  - The potential for third party monitoring, interception and substitution
  - Loss of locational accuracy if the DNS is used to perform content steering for CDNs

DNS Privacy

The response to the issues of DNS over UDP has been the development of DNS stub-to-recursive connections over encrypted transport sessions

- **DNS over TLS** (DoT) uses DNS over TCP over a persistent TLS session
- **DNS over QUIC** (DoQ) uses DNS over an encrypted QUIC transport session over UDP
- **DNS over HTTPS** (DoH) uses DNS over HTTP/3 (over QUIC) where supported, and DNS over HTTP/2 (over TLS) otherwise

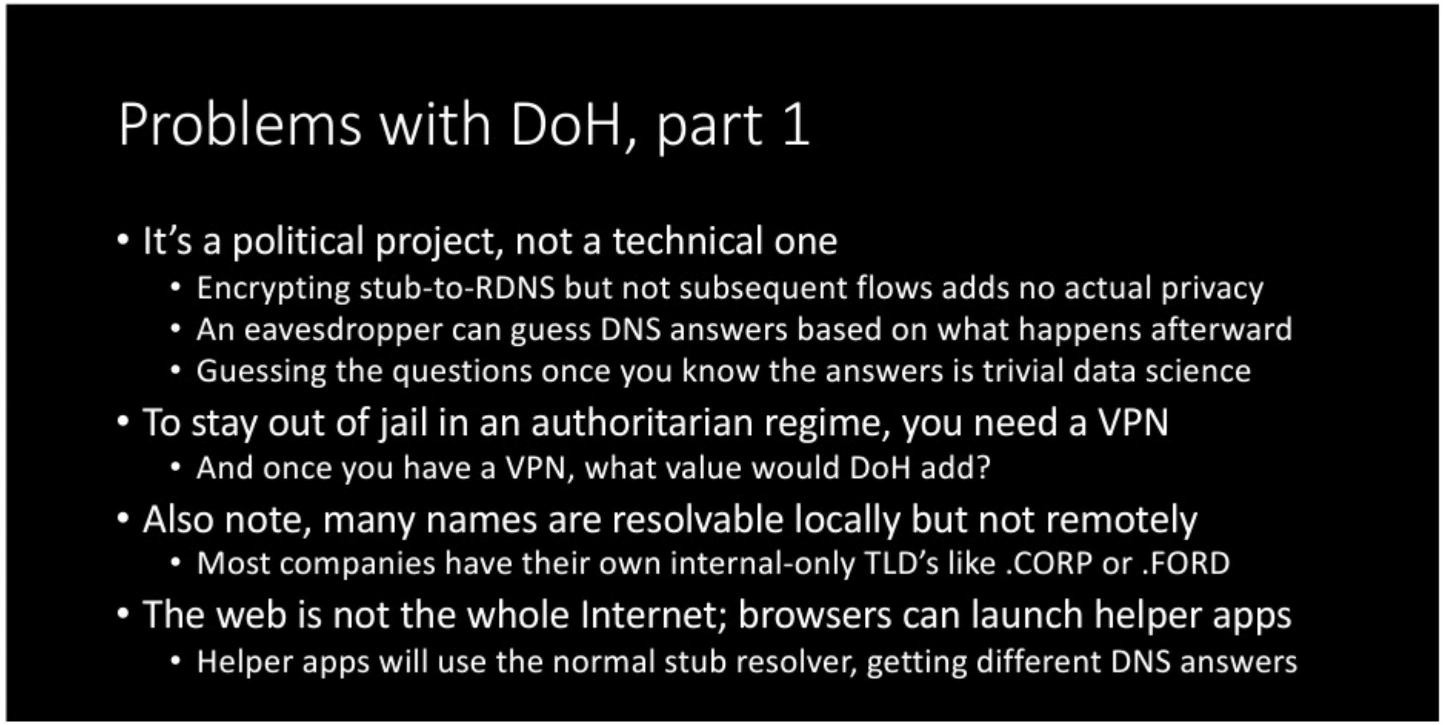
DoT/DoQ/DoH properties

- The stub resolver can authenticate the recursive resolver server identity
  - Which mitigates various forms of session interception
- Session encryption
  - Which mitigates various forms of payload tampering
- No Head of Line Blocking (in DNS over HTTPS/3, and DoQ)
- No UDP fragmentation and TCP failover issues
- Long-held stub/recursive sessions and TCPFO can amortise the encryption and session overheads over many queries
  - Which means it’s not much different to UDP in terms of per query overheads

Why DoH in particular?

- DoH is simply DNS queries and responses packaged with an HTTP header, so why bother? Why not just use DoT or DoQ?

One view:



Problems with DoH, part 1

- It's a political project, not a technical one
  - Encrypting stub-to-RDNS but not subsequent flows adds no actual privacy
  - An eavesdropper can guess DNS answers based on what happens afterward
  - Guessing the questions once you know the answers is trivial data science
- To stay out of jail in an authoritarian regime, you need a VPN
  - And once you have a VPN, what value would DoH add?
- Also note, many names are resolvable locally but not remotely
  - Most companies have their own internal-only TLD's like .CORP or .FORD
- The web is not the whole Internet; browsers can launch helper apps
  - Helper apps will use the normal stub resolver, getting different DNS answers

Paul Vixie, DNS Wars: Episode IV, NANOG 85, 2019

Why DoH?

- DoH is simply DNS queries and responses packaged with an HTTP header, so why bother? Why not just use DoT or DoQ?
- But maybe its more than a political project:
  - Because it sits alongside all other HTTPS traffic on TCP port 443 (HTTP/2) and UDP port 443 (HTTP/3) and is harder for network level isolation of DNS traffic
  - Because generic HTTP caching controls can be used to enable or disable the use of HTTP caching
  - Because HTTP/2 and HTTP/3 includes support for reordering, parallelism, priority and header compression
  - Because applications need not use the local stub DNS resolver and can direct DoH queries to a recursive resolver of its own choice
  - Because HTTP/2 and HTTP/3 includes “Server Push”

"Server Push"?

- RFC 7540: **Hypertext Transfer Protocol Version 2 (HTTP/2)**

```
8.2.  Server Push

   HTTP/2 allows a server to pre-emptively send (or "push") responses
   (along with corresponding "promised" requests) to a client in
   association with a previous client-initiated request.  This can be
   useful when the server knows the client will need to have those
   responses available in order to fully process the response to the
   original request.
```

But in DoH this means…

8.2. Server Push

HTTP/2 allows a server to pre-emptively send (or "push") DNS responses (along with corresponding "promised" DNS requests) to a client in association with a previous client-initiated request. This can be useful when the server knows the client will need to have those DNS responses available in order to fully process the response to the original request.

What about HTTP/3?

- Yes – **draft-ietf-quic-http**

4.4. Server Push

Server push is an interaction mode that permits a server to push a request-response exchange to a client in anticipation of the client making the indicated request. This trades off network usage against a potential latency gain. HTTP/3 server push is similar to what is described in Section 8.2 of [HTTP2], but uses different mechanisms.

Each server push is assigned a unique Push ID by the server. The Push ID is used to refer to the push in various contexts throughout the lifetime of the HTTP/3 connection.

So this means…

- When a server sends a response to an HTTP request it can also push unrequested DNS responses
- This allows the user application to use these DNS resolution outcomes immediately and bypass DNS resolution delays
  - It’s faster
- The user is not making these resolution queries, and is not generating meta data within the DNS
  - It has some privacy benefits

But …

- How do you know that the server is pushing the “truth” when it provides these DNS answers?
- The secure transport means that tampering is challenging, but the user should still validate these responses (assuming that they are DNSSEC-signed in the DNS)
- Which would mean that the user still has to chase down the DNSSEC validation chain, and most of the the original speedup advantages are lost
- Or maybe not…
  - The server could also push the collection of DNSSEC validation responses to the client
  - The server could also repackage these responses into a RFC7901 EDNS0 Chain Response, attached to the original response

DNSSEC-Validated DNS data

- DNSSEC validation provides the user with assurance that the data is:
  - Authentic
  - Current
  - Complete (for each RRType)
- If that’s the case then why does it matter how the stub learned the data?
  - It could be a DNS query / response transaction
  - It could be via a server push over DoH
- DNSSEC validation is providing the assurance that the data is usable

What if the DNS response is unsigned?

- Er, um, er…
- It’s probably best to discard it!
  - You have no idea how the server obtained the DNS data in the first place
  - You don’t know how current the data is
  - You really don’t know if the server is trying to deceive you
  - And you have no idea who you are implicitly trusting if you use the data

What can we say about Resolverless DNS?

- It gives HTTP-based applications and services far more control over the quality of the user experience
  - It allows the server to pre-provision the client with DNS data that is likely to be useful in the context of the application
  - It allows the client side application to perform rapid DNSSEC validation without relying on stub resolver capabilities and settings
  - It can replace UDP-based timers, query retries, fragmentation and TCP switching with server-to-client provision
  - It operates over a secured connection with an authenticated server

- As long as the DNS data is DNSSEC-signed

What about unsigned DNS data?

- Forget it!
- Resolverless DNS responses of unsigned DNS data just opens up more potential vulnerabilities with little in the way of reasonable mitigation

Where is this leading?

The changing economics of the Internet

- The shift to advertiser-funded content and service has sucked the revenue base from access and common infrastructure
- Internet infrastructure is a commodity-based activity which resists innovation
- The incentives to innovate lie in the application and service layers
- Infrastructure is under continued pressure to achieve further efficiencies and there is a consequent pressure to scale up and centralise

The DNS is caught up in this, and innovation in the DNS is extremely challenging to get adoption these days

Does Resolverless DNS have a chance?

Is there a role for Resolverless DNS?

Perhaps

- But I would suggest it necessarily assumes DNSSEC
- Which, at present, is a tough assumption
  - Because few zones are signed
    - Because DNSSEC signatures tend to create larger responses, which is a problem in UDP DNS
    - Because advanced zone signing a zone is infeasible for very large zones
    - And signing on the fly can be fragile
    - In summary, few zones are DNSSEC are signed is because its unreliable and expensive and the benefits do not seem to offset against the additional risks
  - And few resolvers validate
    - Because it takes time
    - And stresses out UDP DNS

What if we could dream?

- So far we’ve thought of the DNS as a constant
  - It’s the same answer whether its delivered over UDP, DoT, DoQ or DoH
- But what if we could think about a DNS whose properties change depending on the mode of delivery of responses?
  - What if we could work with a zone appears to be DNSSEC-signed when we use resolverless DoH, but the same zone appears to be unsigned in DNS over UDP?
    - i.e. there is a DNSSEC validation path that exists only when it’s chained to a response, but it’s not visible when specifically queried as a sequence of DNS transactions?
- Or is this a completely fanciful notion?

Why is all this interesting?

We have spent a huge amount of effort over the last decade trying to make the Internet faster:

- We’ve been deploying CDNs to replicate content and services and bring them closer to users
- We’ve been deploying non-blocking transport protocols (such as QUIC) to exploit parallelism
- We’ve been tuning TCP and network behaviour to create more efficient and faster network transactions
- We’ve been packing more information in the DNS to make service startup faster (SVC and HTTPS records)

Why is all this interesting?

- Yet the DNS is still:
  - A massive time penalty
  - A significant privacy leak
  - A consistent source of failure
- Resolverless DNS won’t fix all the DNS all at once
  - But it can hand a significant amount of control over application and service quality back to these HTTPS-based applications and services
  - And it’s a whole lot faster!
- And for those reasons it’s a very interesting step in the possible evolution of the DNS

Thanks!