

# A Question of Protocol

Geoff Huston  
APNIC

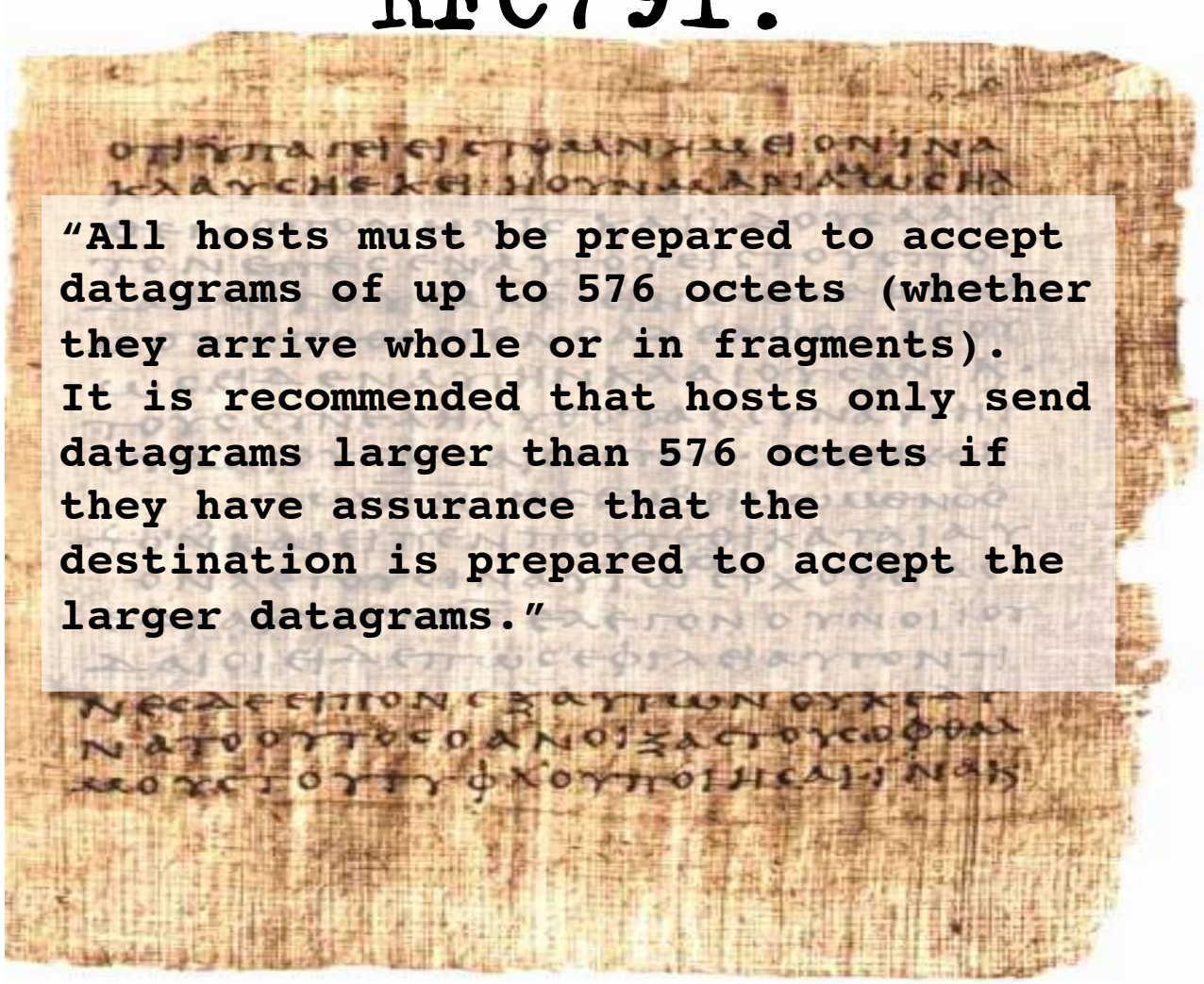


Originally there was  
RFC791:

# Originally there was RFC791:

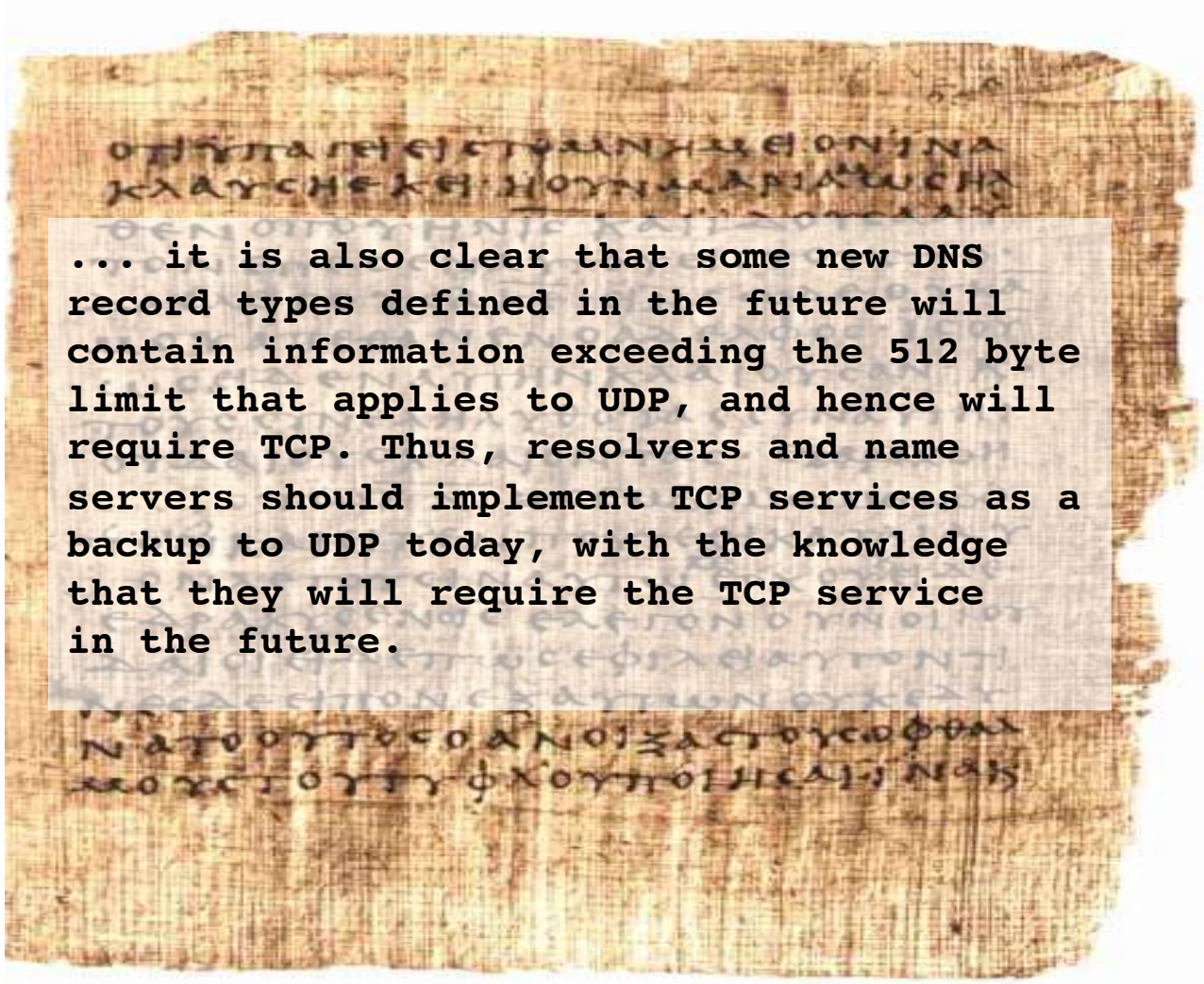


# Originally there was RFC791:

A piece of ancient papyrus with a white rectangular overlay containing text. The papyrus has a textured, fibrous appearance and is covered in faint, illegible markings that resemble ancient script. The white overlay is centered on the papyrus and contains a quote in a monospaced font.

"All hosts must be prepared to accept datagrams of up to 576 octets (whether they arrive whole or in fragments). It is recommended that hosts only send datagrams larger than 576 octets if they have assurance that the destination is prepared to accept the larger datagrams."

# Then came RFC1123:

A fragment of an ancient papyrus scroll with a semi-transparent text box overlaid. The scroll is made of yellowish-brown papyrus with visible fibers. The text on the scroll is written in ancient Greek, appearing as dark ink on the textured surface. The text box is a light beige rectangle with a thin black border, containing modern English text in a monospaced font.

... it is also clear that some new DNS record types defined in the future will contain information exceeding the 512 byte limit that applies to UDP, and hence will require TCP. Thus, resolvers and name servers should implement TCP services as a backup to UDP today, with the knowledge that they will require the TCP service in the future.

# Then came RFC1123:

... it is also clear that some new DNS record types defined in the future will contain information exceeding the 512 byte limit that applies to UDP, and hence will require TCP. Thus, resolvers and name servers should implement TCP services as a backup to UDP today, with the knowledge that they will require the TCP service in the future.

is that a "SHOULD", or a mere "should"?

# Hang on...

RFC 791 said 576 octets, yet RFC 1123 reduces this even further to 512 bytes

What's going on?

An IPv4 UDP packet contains:

- 20 bytes of IP header

- $\leq 40$  bytes of IP options

- 8 bytes of UDP header

- payload

The IP header is between 28 and 68 bytes

All hosts must accept a 576 byte IP packet, which implies that the maximum UDP payload that all hosts will accept is 512 bytes

# The original DNS model

If the reply is  $\leq 512$  bytes, send a response over UDP

If the reply is  $> 512$  bytes, send a response over UDP, but set the TRUNCATED bit in the DNS payload

- Which should trigger the client to re-query the server using TCP



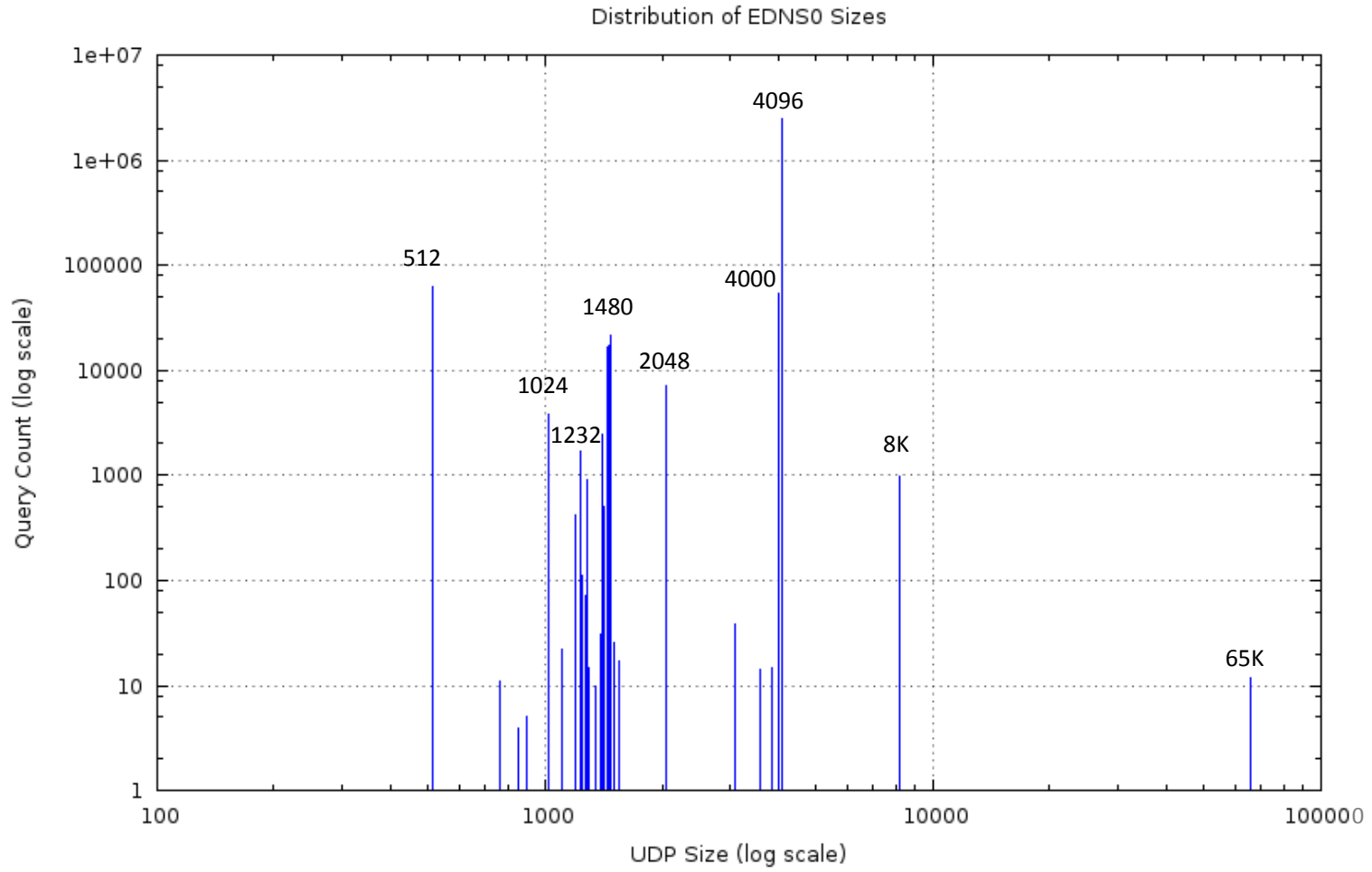
# Then came EDNS0

## RFC2671:

4.5. The sender's UDP payload size (which OPT stores in the RR CLASS field) is the number of octets of the largest UDP payload that can be reassembled and delivered in the sender's network stack. Note that path MTU, with or without fragmentation, may be smaller than this.

The sender can say to the resolver: “It’s ok to send me DNS responses using UDP up to size <xxx>. I can handle packet reassembly.”

# Aside: Offered EDNS0 Size Distribution



# Aside: Offered EDNSO Size Distribution

512	62977	1420	513	
768	11	1440	10443	
850	4	1450	16332	
900	5	1452	3605	← IPv6?? 1500 - 48
1024	3857	1460	17387	
1100	22	1472	1933	
1200	416	1480	21225	← ?? 1500 - 20
1232	1706	1500	26	
1252	112	1550	17	
1272	71	2048	6984	
1280	906	3072	38	← ?? IPv6
1300	15	3584	14	
1352	10	3839	15	
1392	31	4000	54492	
1400	2431	4096	2500352	← RFC 6891
1410	1291	8192	981	
1412	209	65535	12	

# What if...

One were to send a small query in UDP to a DNS resolver with:

- EDNS0 packet size set to a large value

- The IP address of the intended victim as the source address of the UDP query

- A query that generates a large response in UDP

  - ISC.ORG IN ANY, for example

You get a 10x – 100x gain!

Mix and repeat with a combination of a bot army and the published set of open recursive resolvers (of which there are currently some 28 million!)

Which leads to...



# Possible Mitigations...?

- 1) Get everyone to use BCP38

# Possible Mitigations...?

- 1) Get everyone to use BCP38
- 2) Use a smaller EDNS0 max size

# Possible Mitigations...?

- 1) Get everyone to use BCP38
- 2) Use a smaller EDNS0 max size
- 3) Selectively push back with TC=1



# Possible Mitigations...?

- 1) ~~Get everyone to use BCP38~~
- 2) Use a smaller EDNS0 max size
- 3) Selectively push back with TC=1

So lets look at 2) & 3):

This would then force the query into TCP

And the TCP handshake does not admit source address spoofing

# Could this work?

How many customers use DNS resolvers that support TCP queries?

– Lets find out with an experiment:

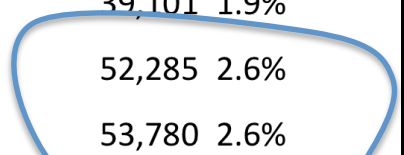
- Turn down the EDNS0 size limit on an authoritative server to 512 bytes
- Enlist a large number of clients to fetch a collection of URLs:
  - Short DNS name, unsigned (fits in a 512 byte UDP response)
  - Short DNS name, DNSSEC-signed
  - Long DNS name, unsigned
  - Long DNS name, DNSSEC-signed

# Results

DNS Name	UDP Queries	Truncated UDP Responses	TCP responses	Truncated UDP to TCP Fail
Short, unsigned	2,029,725	2	6	0
Short, signed	2,037,563	1,699,935 83.4%	1,660,754 81.5%	39,101 1.9%
Long, unsigned	2,023,205	2,021,212 99.9%	1,968,927 97.3%	52,285 2.6%
Long, signed	2,033,535	2,032,176 99.9%	1,978,396 97.3%	53,780 2.6%

# Results

DNS Name	UDP Queries	Truncated UDP Responses	TCP responses	Truncated UDP to TCP Fail
Short, unsigned	2,029,725	2	6	0
Short, signed	2,037,563	1,699,935 83.4%	1,660,754 81.5%	39,101 1.9%
Long, unsigned	2,023,205	2,021,212 99.9%	1,968,927 97.3%	52,285 2.6%
Long, signed	2,033,535	2,032,176 99.9%	1,978,396 97.3%	53,780 2.6%



# Results

To get to the long name with a >512 byte response we used *cnames*:

4a9c317f.4f1e706a.6567c55c.0be33b7b.2b51341.a35a853f.59c4df1d.3b069e4e.87ea53bc.2b4cfb4f.987d5318.fc0f8f61.3cbe5065.8d9a9ec4.1ddfa1c2.4fee4676.1ffb7fcc.ace02a11.a3277bf4.2252b9ed.9b15950d.db03a738.dde1f863.3b0bf729.04f95.z.dotnxdomain.net.

CNAME

33d23a33.3b7acf35.9bd5b553.3ad4aa35.09207c36.a095a7ae.1dc33700.103ad556.3a564678.16395067.a12ec545.6183d935.c68cebfb.41a4008e.4f291b87.479c6f9e.5ea48f86.7d1187f1.7572d59a.9d7d4ac3.06b70413.1706f018.0754fa29.9d24b07c.04f95.z.dotnxdomain.net

33d23a33.3b7acf35.9bd5b553.3ad4aa35.09207c36.a095a7ae.1dc33700.103ad556.3a564678.16395067.a12ec545.6183d935.c68cebfb.41a4008e.4f291b87.479c6f9e.5ea48f86.7d1187f1.7572d59a.9d7d4ac3.06b70413.1706f018.0754fa29.9d24b07c.04f95.z.dotnxdomain.net. A 199.102.79.187,

# Results

To get to the long name with a >512 byte response we used *cnames*

Are these *cnames* causing a higher dropout rate?

We re-ran the experiment with a mangled DNS authoritative name server that had a lowered max UDP response size of 275 bytes, which allowed us to dispense with the *cname* construct

# Results (2)

DNS Name	UDP Queries	Truncated UDP Responses	TCP responses	Truncated UDP to TCP Fail
Short, unsigned	936,007	0	3	3
Short, signed	936,116	935,990 100.0%	916,251 97.9%	19,739 2.1%
Long, unsigned	920,613	920,483 100.0%	896,953 97.4%	23,530 2.6%
Long, signed	934,446	934,330 100.0%	910,757 97.5%	23,573 2.5%

it looks like the cname construct is not influencing the results!

# Results

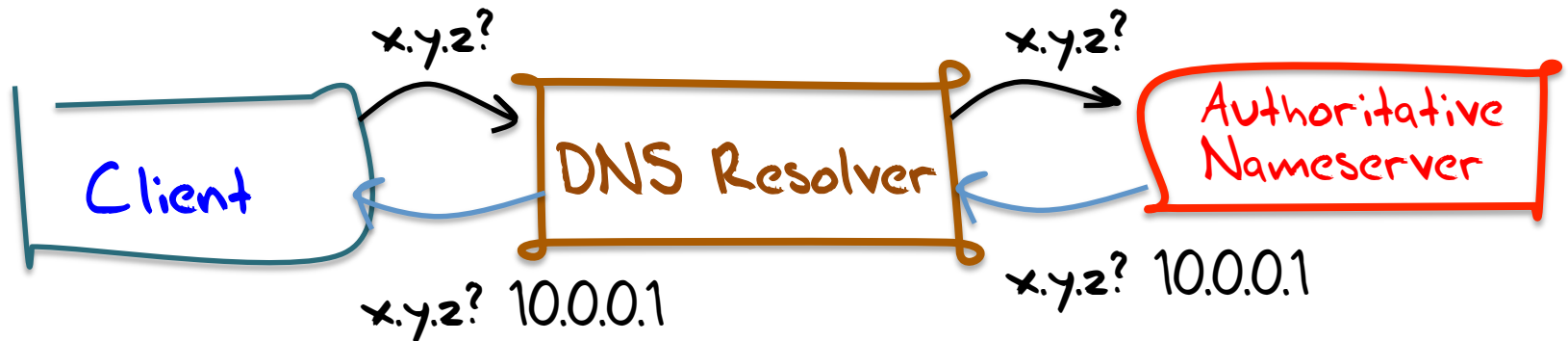
2.6% of clients use a set of DNS resolvers that are incapable of reverting to TCP upon receipt of a truncated UDP response from an authoritative name server

(The failure here in terms of reverting to TCP refers to resolvers at the “end” of the client’s DNS forwarder chain who are forming the query to the authoritative name server)

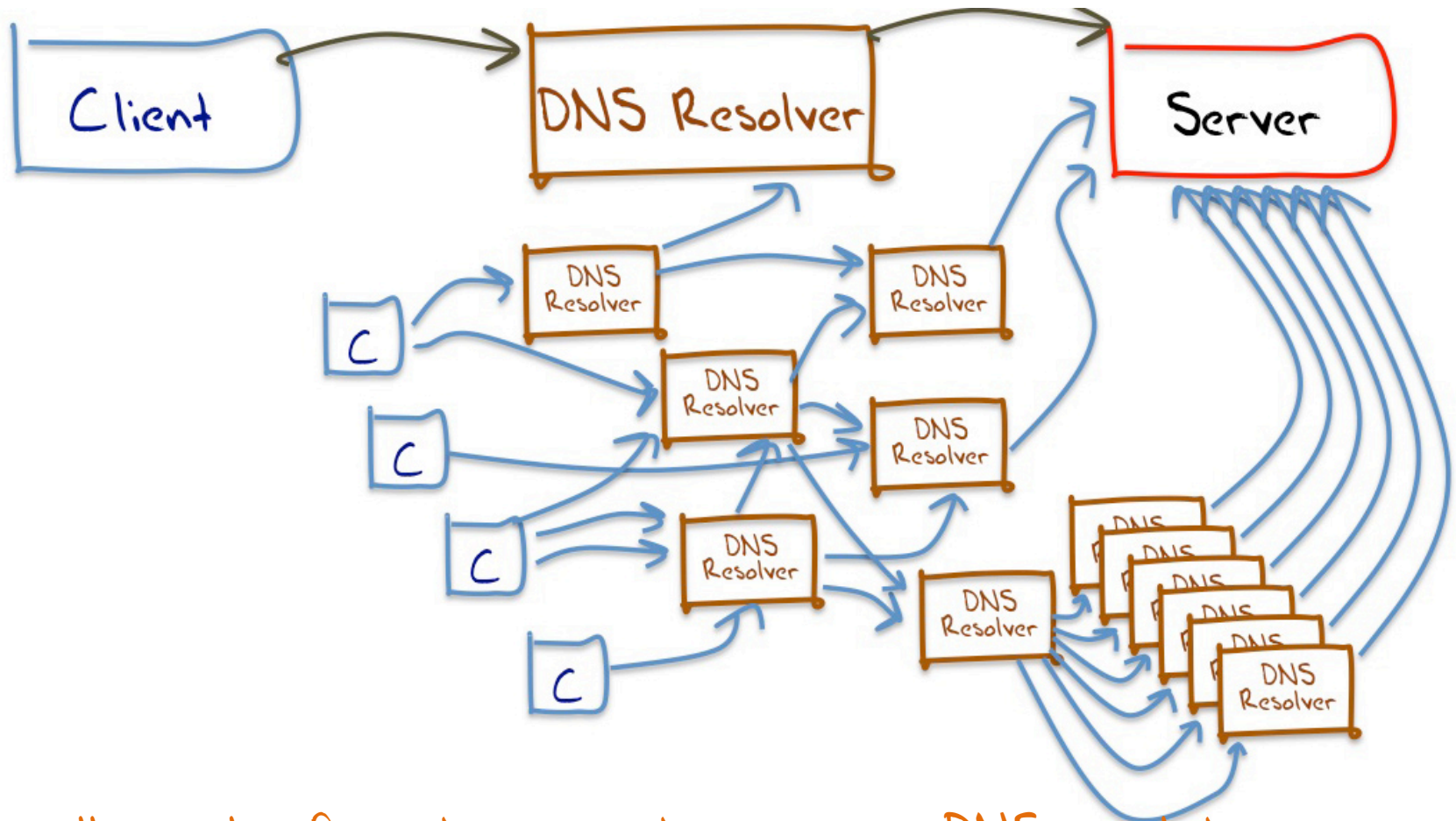


# Aside: Understanding DNS Resolvers is "tricky"

What we would like to think happens in DNS resolution!



# Aside: Understanding DNS Resolvers is "tricky"

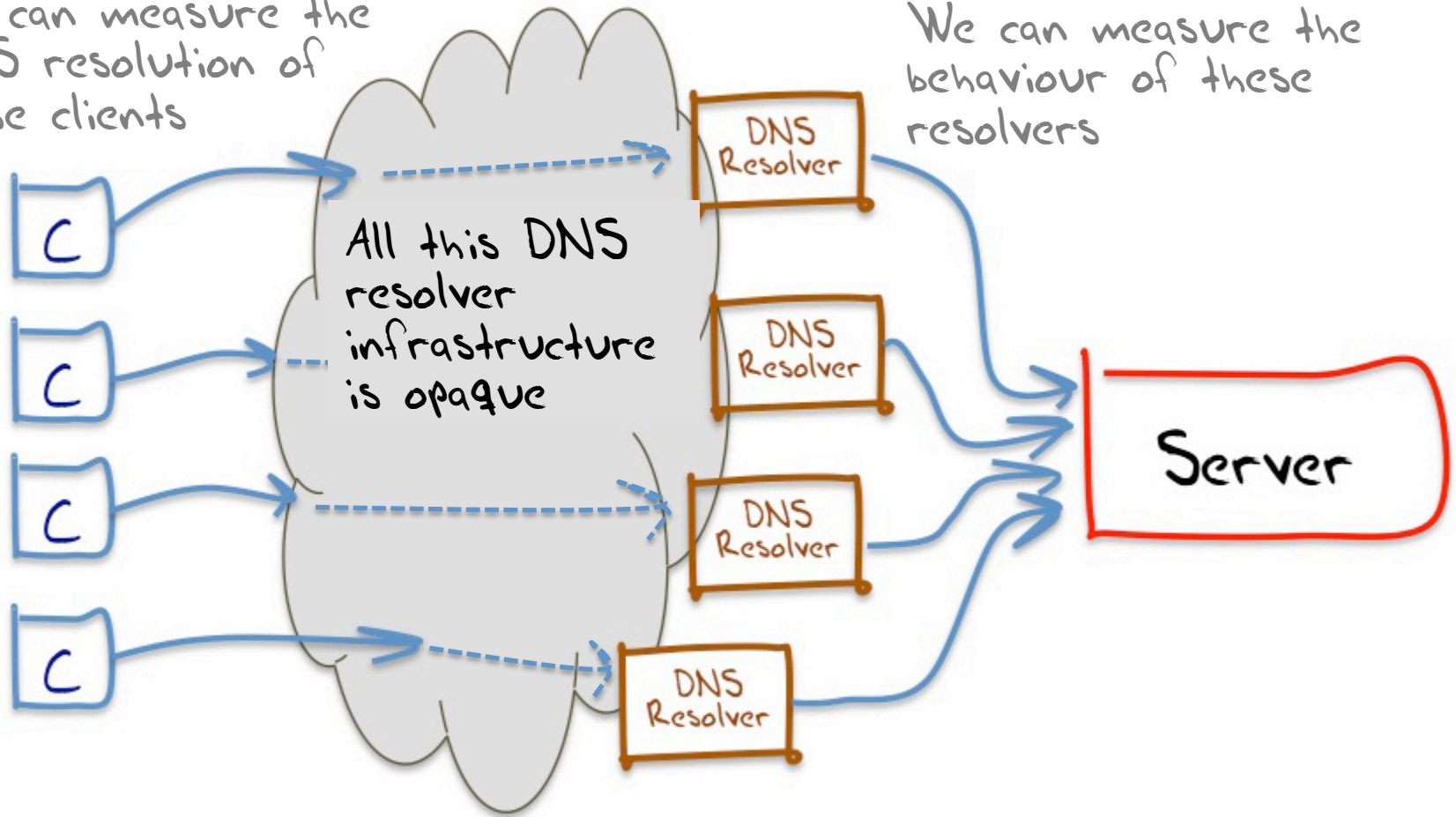


A small sample of what appears to happen in DNS resolution

# Aside: Understanding DNS Resolvers is "tricky"

We can measure the DNS resolution of these clients

We can measure the behaviour of these resolvers



The best model we can use for DNS resolution in these experiments

# Can we say anything about these "visible" resolvers?

## Visible Resolvers

Total Seen: 80,505

UDP only: 13,483

17% of resolvers cannot ask a query in TCP following receipt of a truncated UDP response

6.4% of clients uses these resolvers

3.8% of them failover to use a resolver that can ask a TCP query

2.6% do not

# Can we say anything about these "visible" resolvers?

## Visible Resolvers

Total Seen: 80,505

UDP only: 13,483

**17%** of resolvers cannot ask a query in TCP following receipt of a truncated UDP response

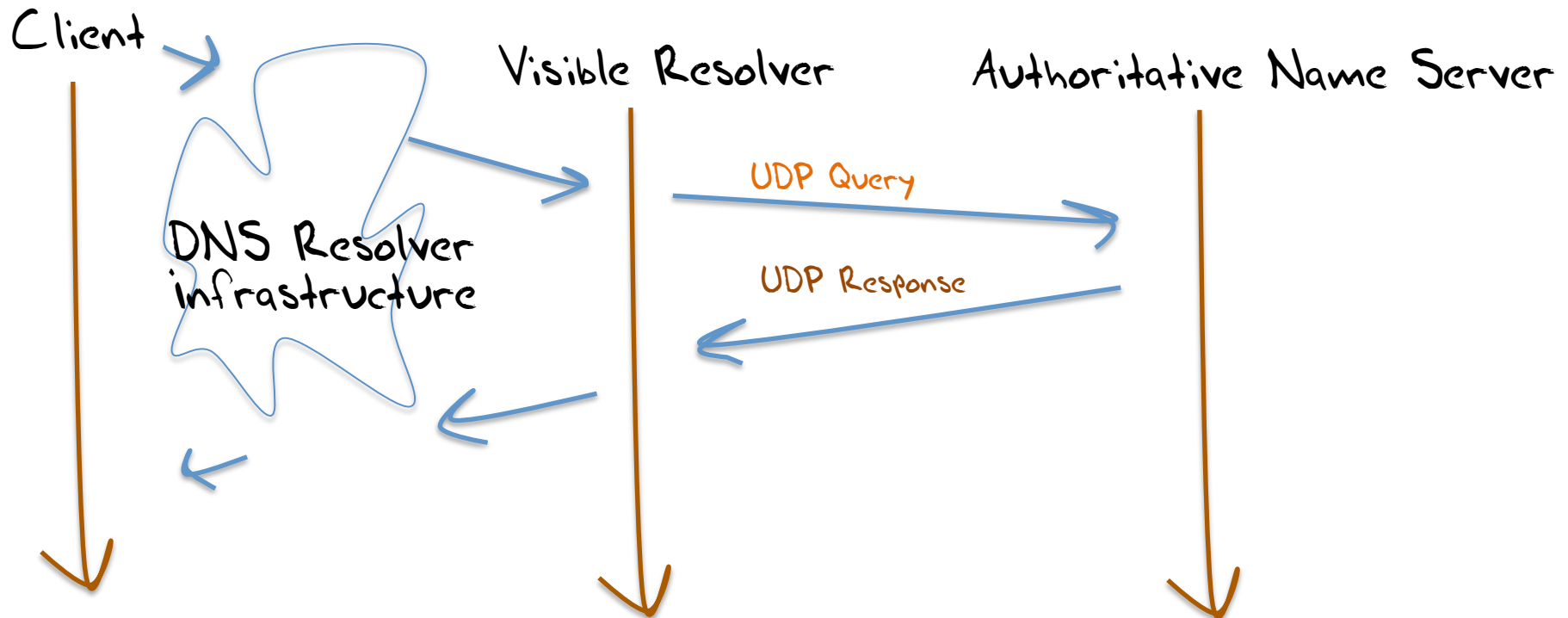
**6.4%** of clients uses these resolvers

3.8% of them failover to use a resolver that can ask a TCP query

2.6% do not

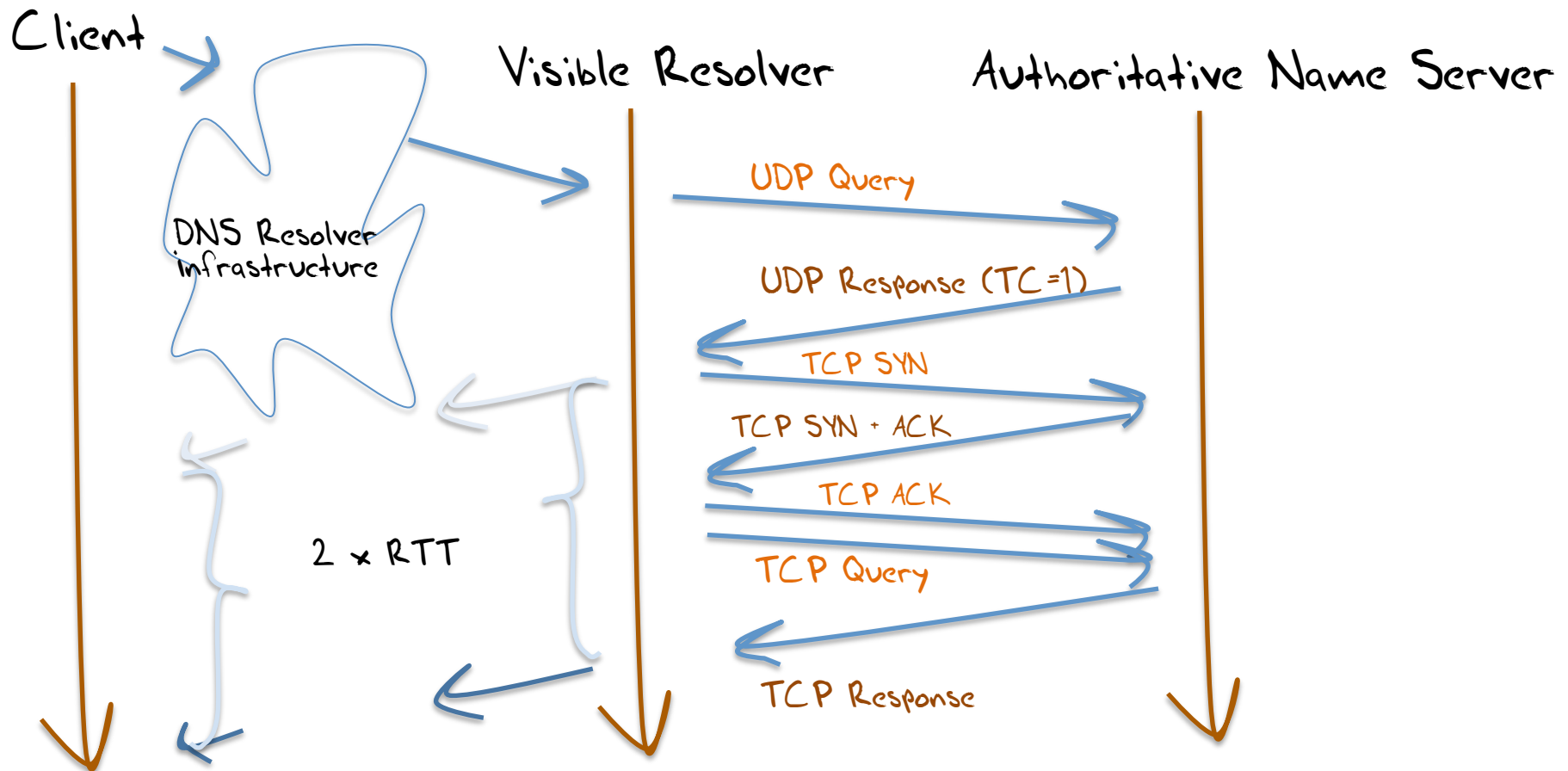
# What about DNS resolution performance?

The theory says:



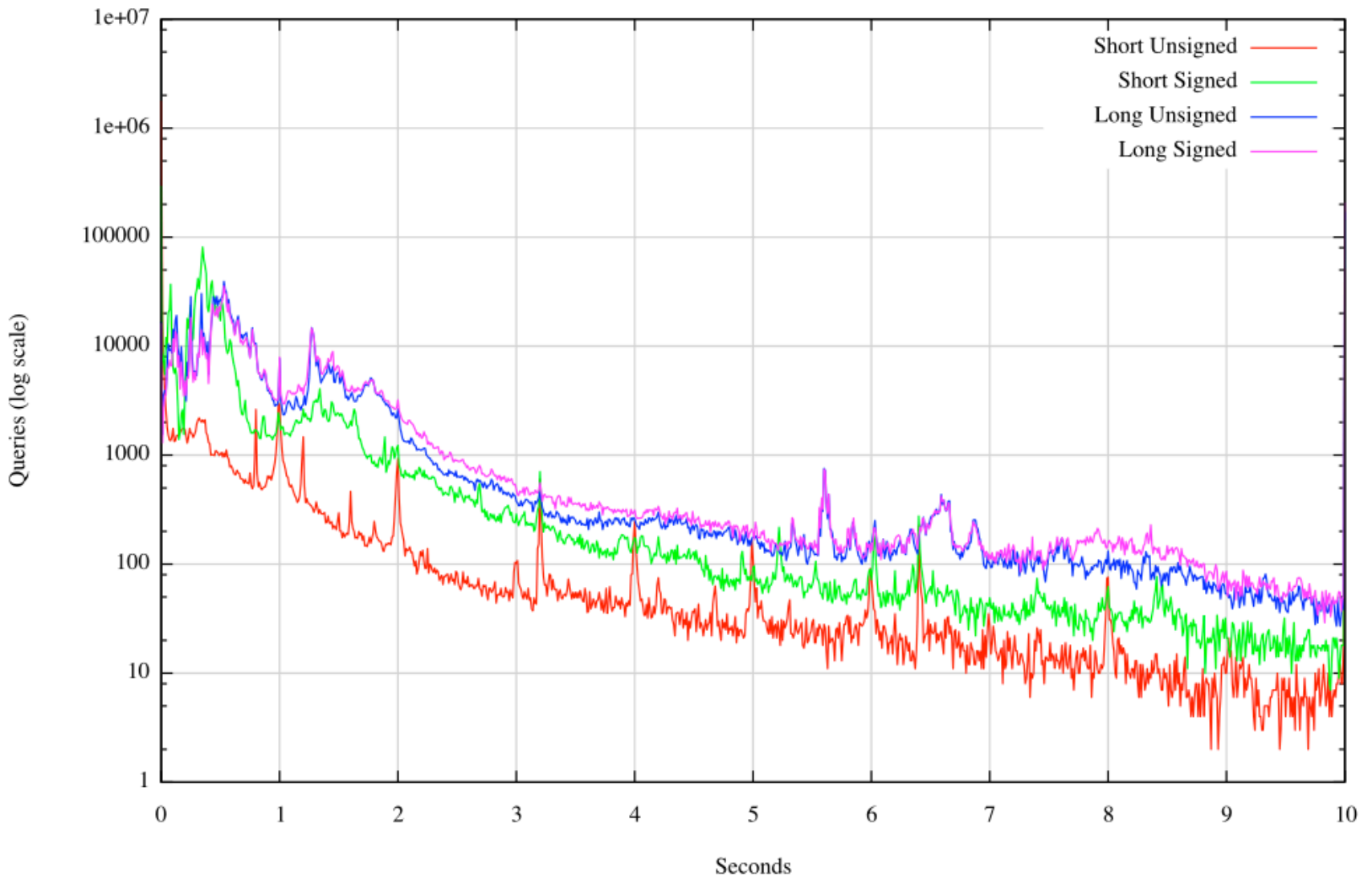
# What about DNS resolution performance?

The theory says:



# Time to resolve a name

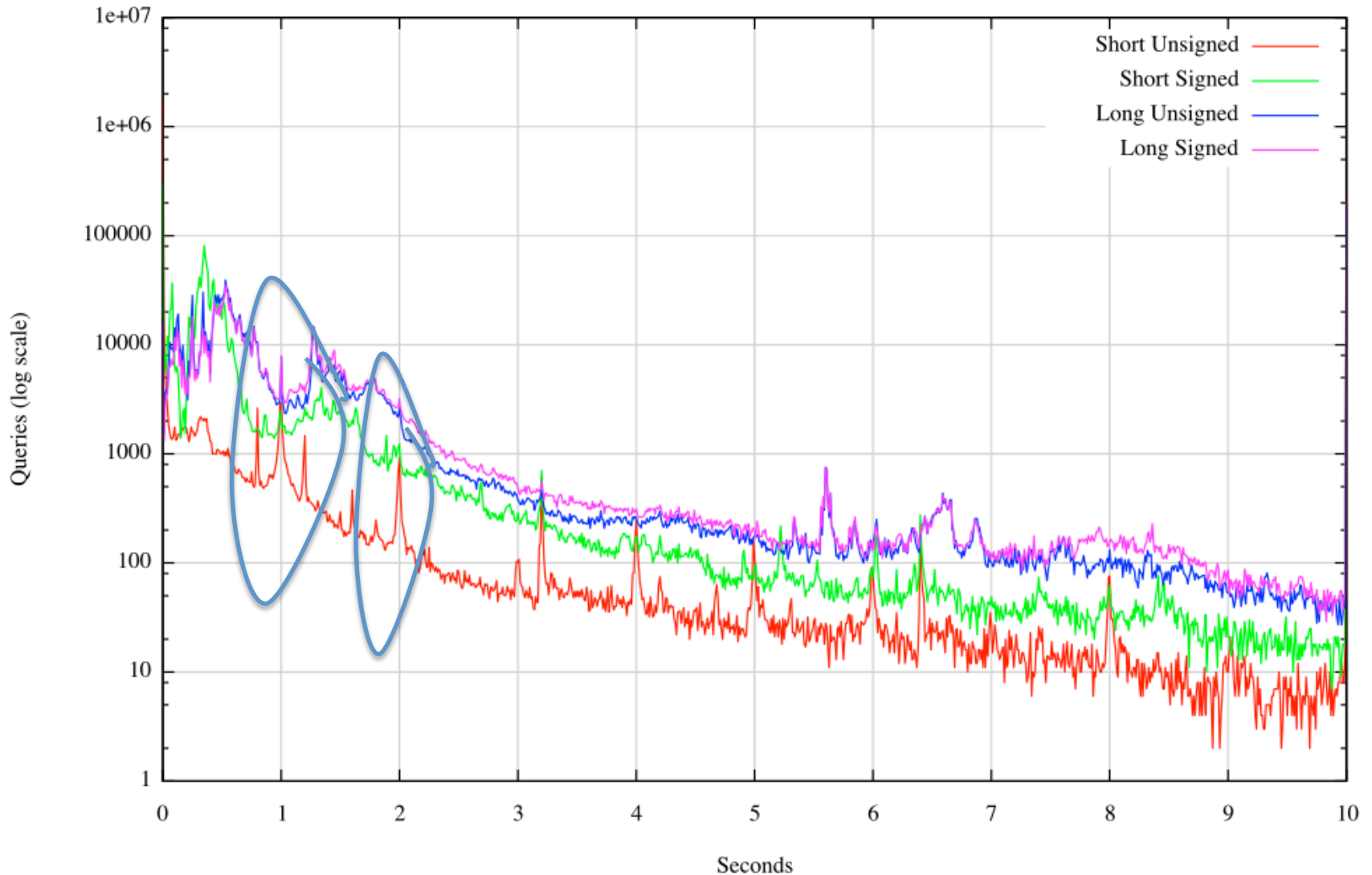
DNS Query Time (At Authoritative Nameserver)





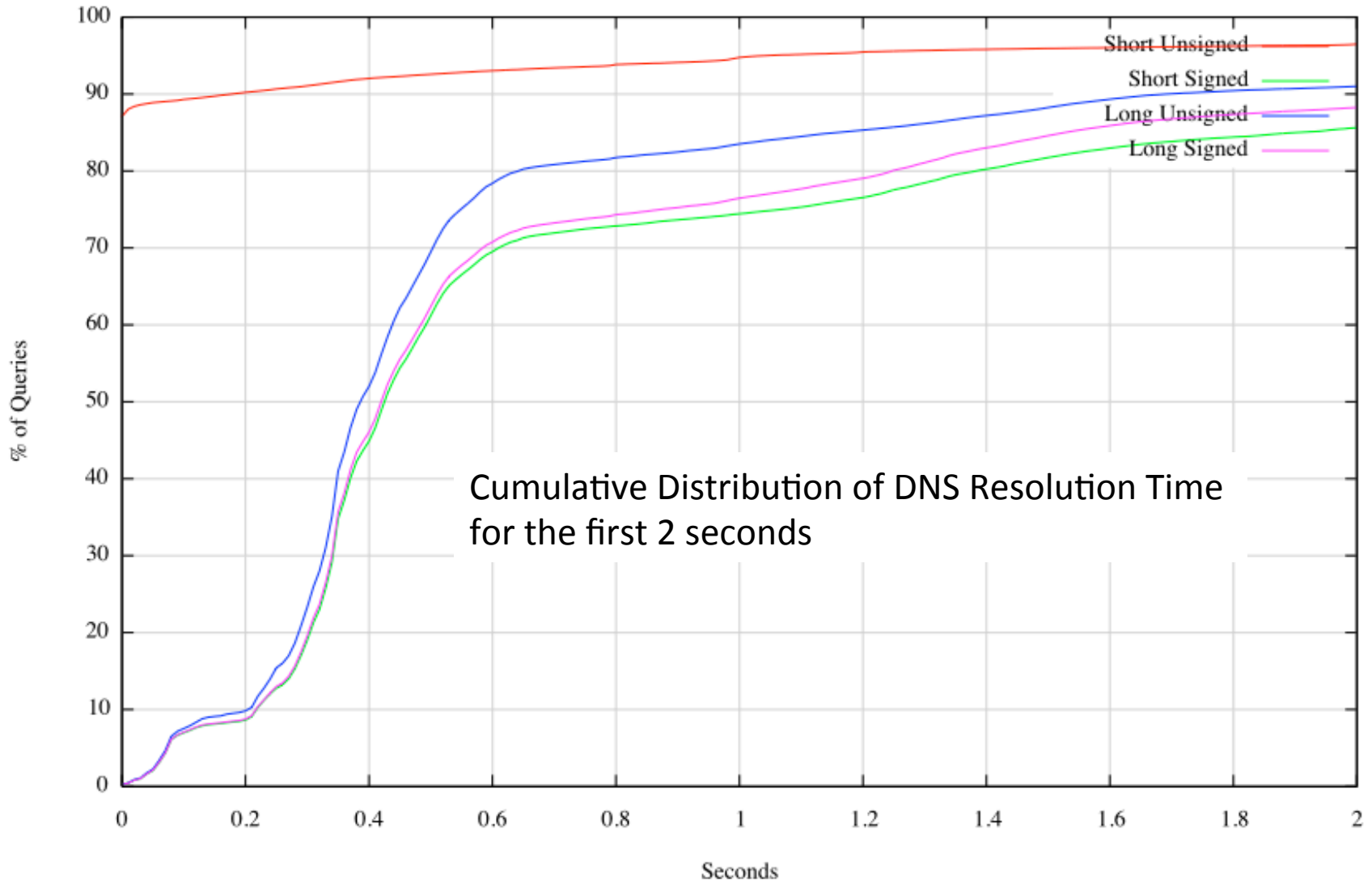
# Time to resolve a name

DNS Query Time (At Authoritative Nameserver)



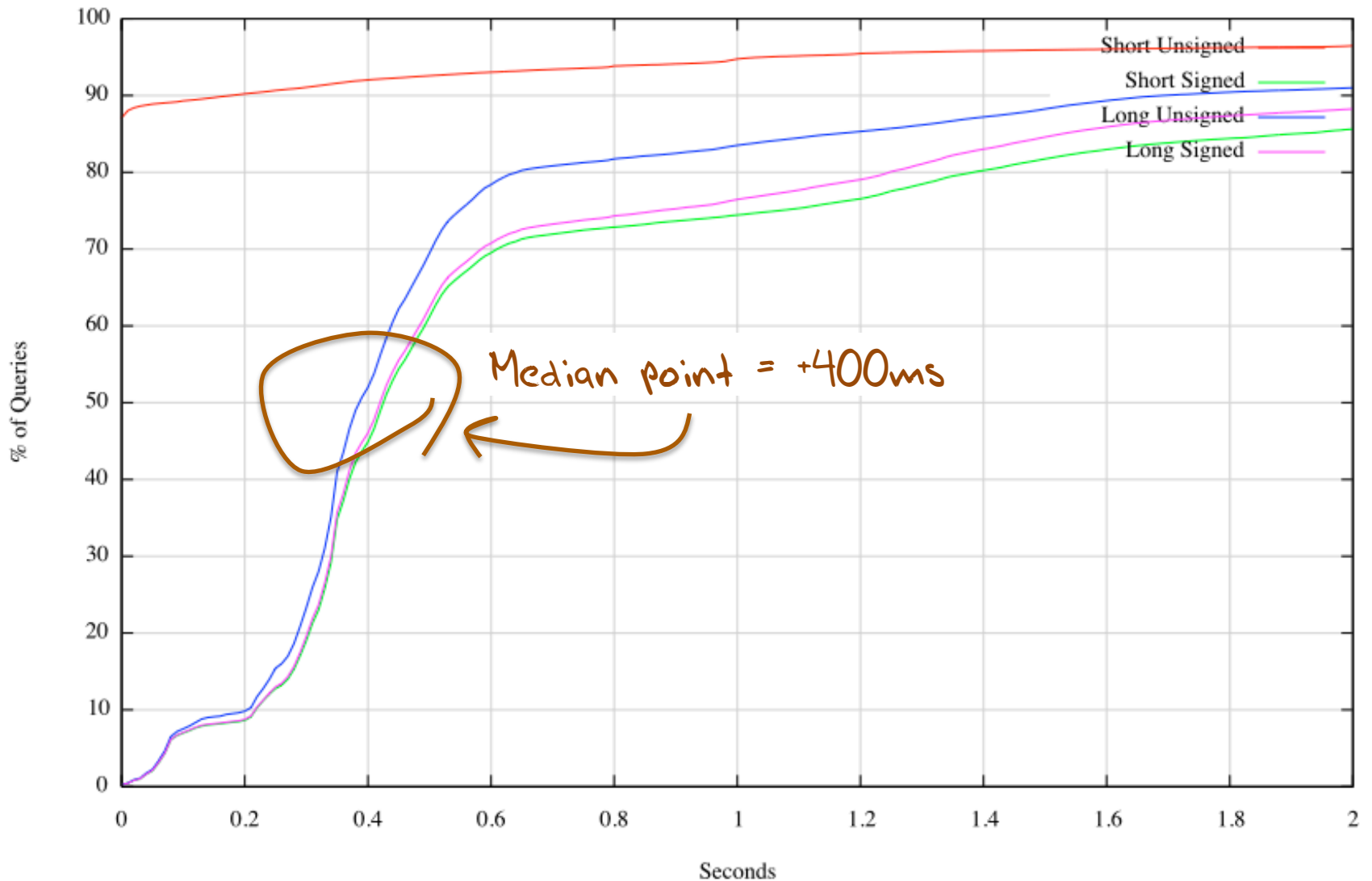
# Time to resolve a name

DNS Query Time (At Authoritative Name Server)



# Time to resolve a name

DNS Query Time (At Authoritative Name Server)



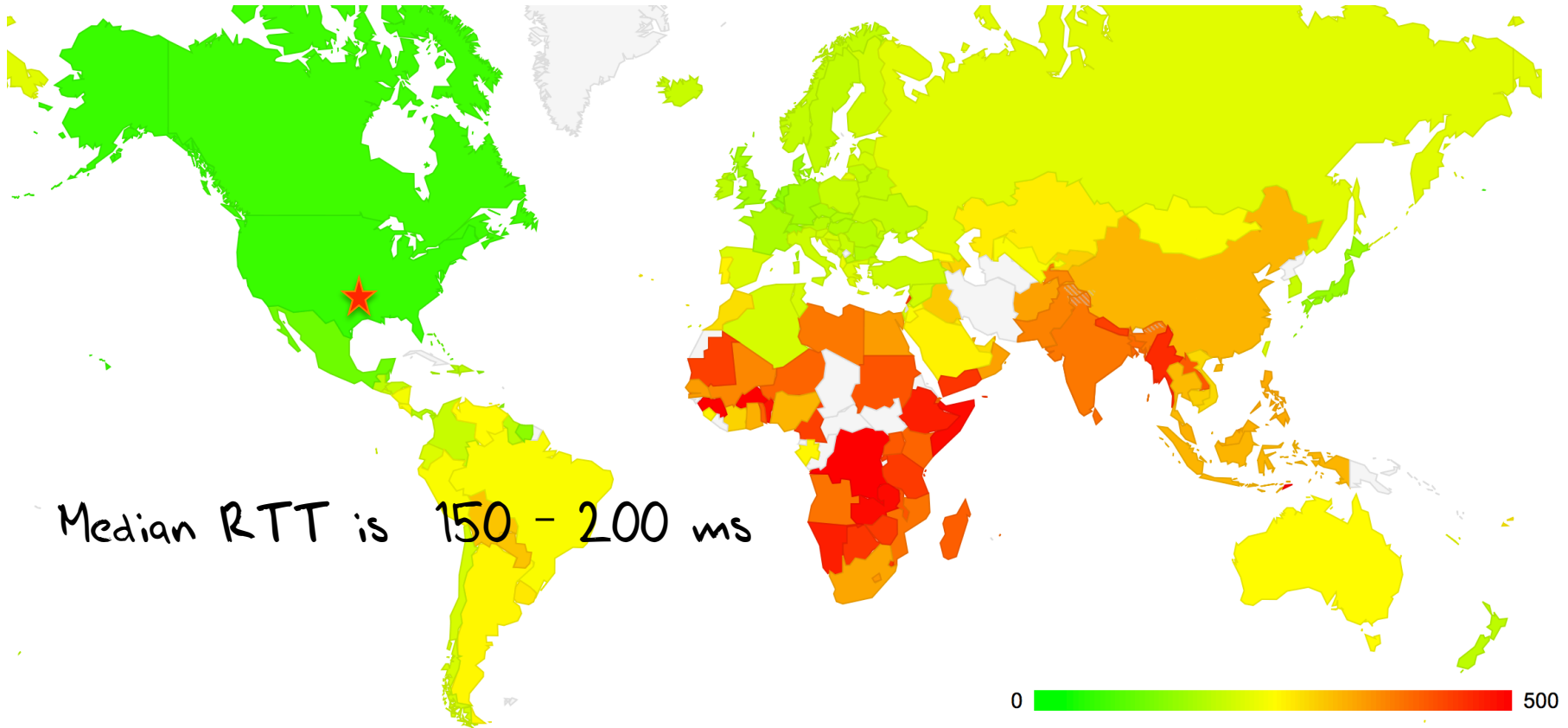
# Time to resolve a name

How does this median value of 400ms relate to the RTT measurements to reach the authoritative name server?

The authoritative name server is located in Dallas, and the initial TCP SYN/ACK exchange can provide an RTT measurement sample

We can geo-locate the resolver IP addresses to get the following RTT distribution map

# Measured RTT Distributions by Country



# DNS over TCP

Around 70% of clients will experience an additional DNS resolution time penalty of 2 x RTT time intervals

However the other 30% experience a longer delay.

- 10% of clients experience a multi-query delay with a simple UDP query response
- 20% of clients experience this additional delay when the truncated UDP response forces their resolver to switch to TCP

# If we really want to use DNS over TCP

Then maybe its port 53 that's the problem for these 17% of resolvers and 20% of the clients

Why not go **all** the way?

How about DNS over XML over HTTP over port 80  
over TCP?



Thanks!