A Study of Privacy and Anonymity in the DNS*

Cesar Ghali, Gene Tsudik, **Christopher A. Wood**

University of California Irvine

DNS OARC – Dallas 2016

*Support provided by Verisign

Pitiful Privacy in the DNS

- Encryption only protects query contents [1,2,3]
- Side channels are prevalent in the protocol [2]:
  - Timing
  - Frequency
  - Response sizes
  - Resolution chains

[1] Bernstein, Daniel J. "DNSCurve: Usable security for DNS." dnscurve. org (2009).
[2] Shulman, Haya. "Pretty bad privacy: Pitfalls of DNS encryption." Proceedings of the 13th Workshop on Privacy in the Electronic Society. ACM, 2014.
[3] DNS-over-HTTPS, Google. https://developers.google.com/speed/public-dns/docs/dns-over-https

Plugging Privacy Holes

- Message padding [size]
- Message interleaving [frequency, time, chains]
- Artificial resolver delays [time]
- Query chaffing [frequency]

But Wait… There’s More

- For privacy, we want to protect the contents of a query from Adv (resolver or stub)
- What about the sources of the queries?
- Can queries reveal information about the origin?

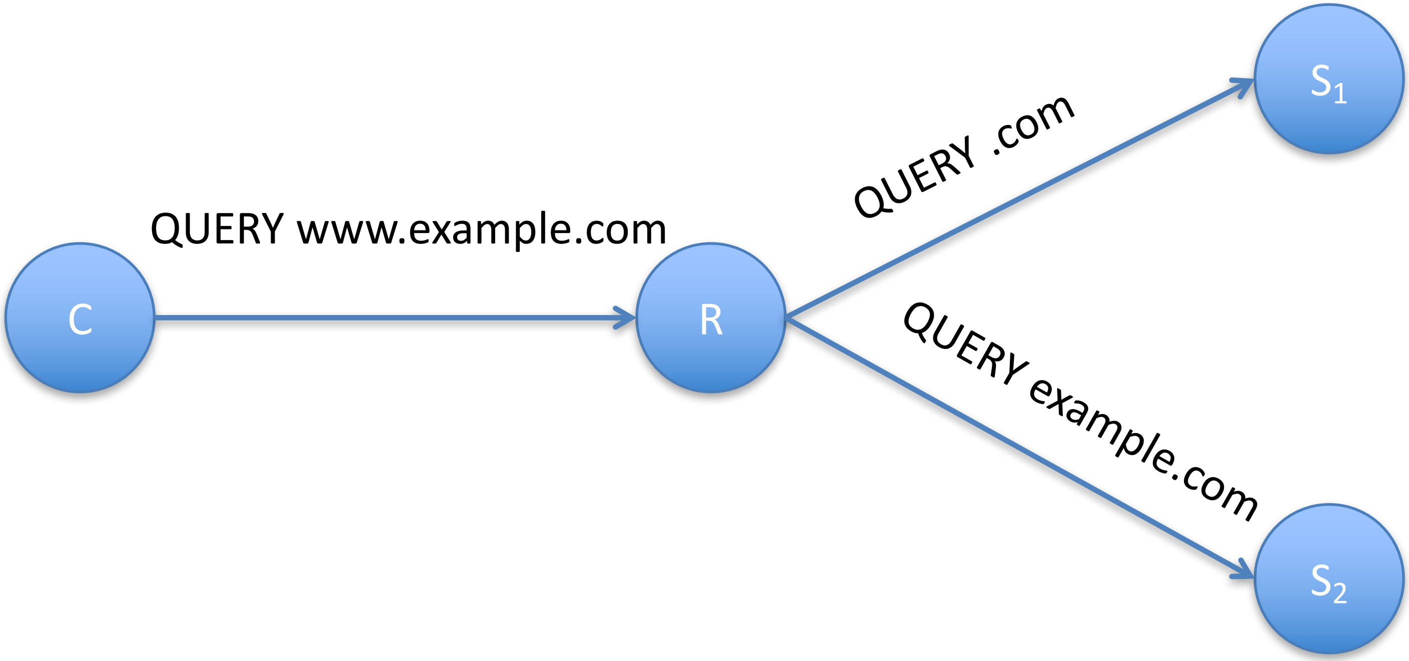
Agenda

- DNS privacy mitigations*
  - Message padding [size]
  - Message interleaving [frequency, time, chains]
  - Artificial resolver delays [time]
- DNS client anonymity
  - Analysis
  - Query chaffing countermeasure [frequency]

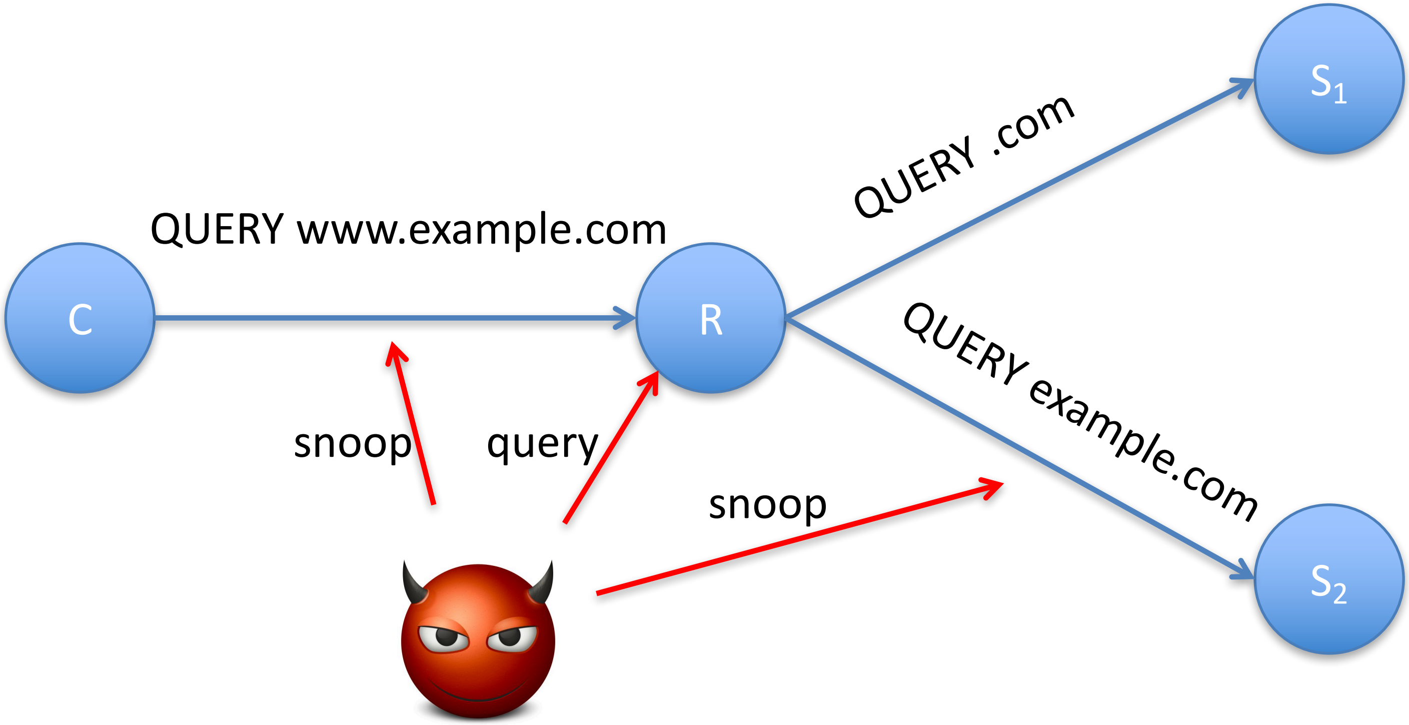
*Strategies implemented in an open source DNS resolver

PRIVACY

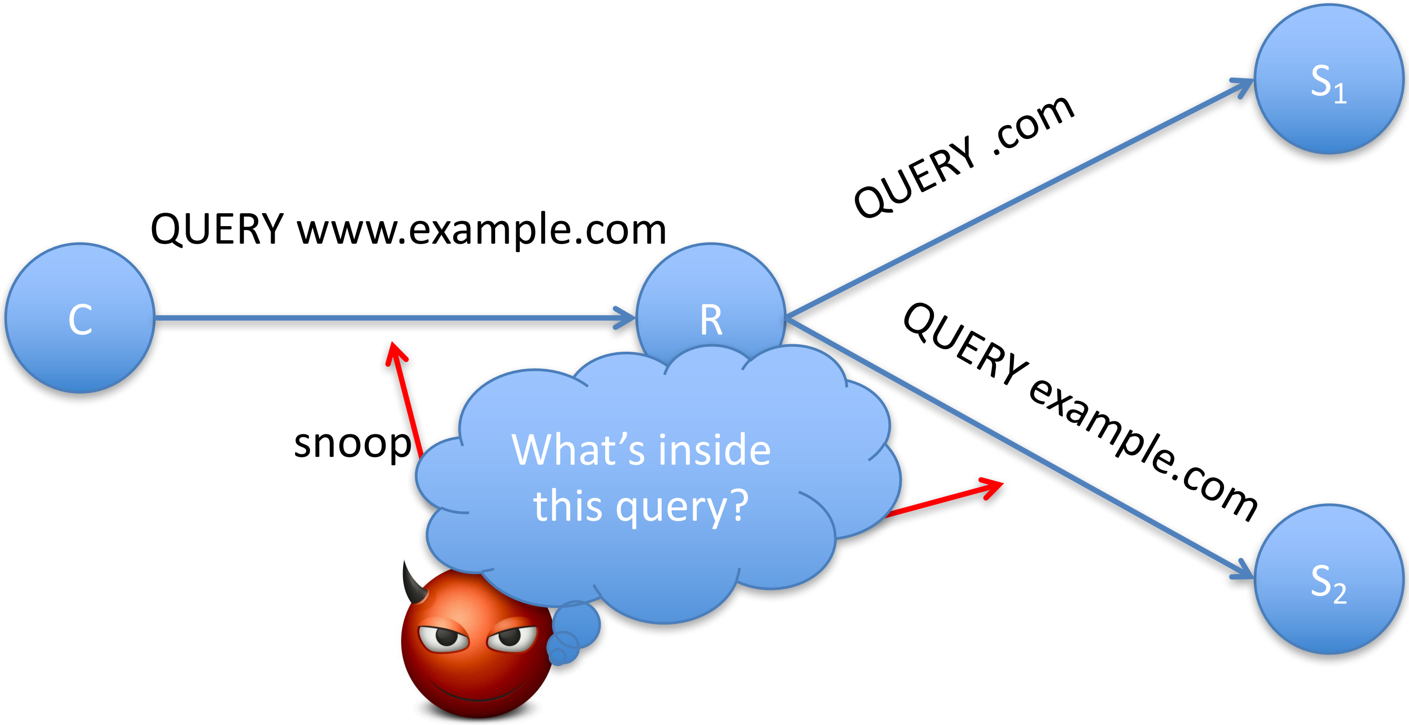
Adversarial Model



Adversarial Model



Adversarial Model



Message Padding
[size]

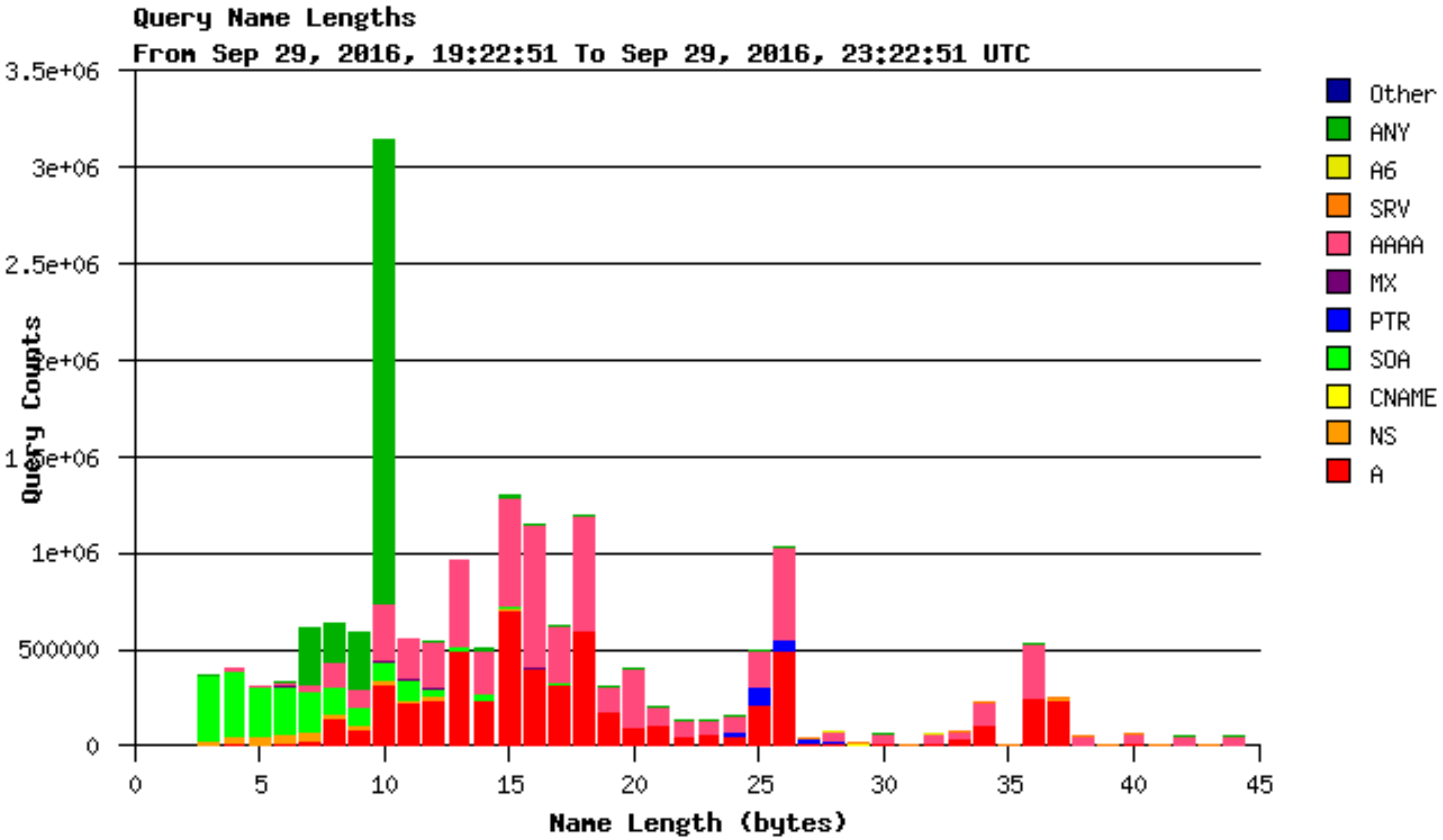
- Ideal requirements:
  - Must fit within UDP packet (or TLS record)
    - What if a request or response exceeds the MTU?
  - Must not be more than what’s necessary
    - What’s the maximum padding length?

EDNS(0) Padding [1]

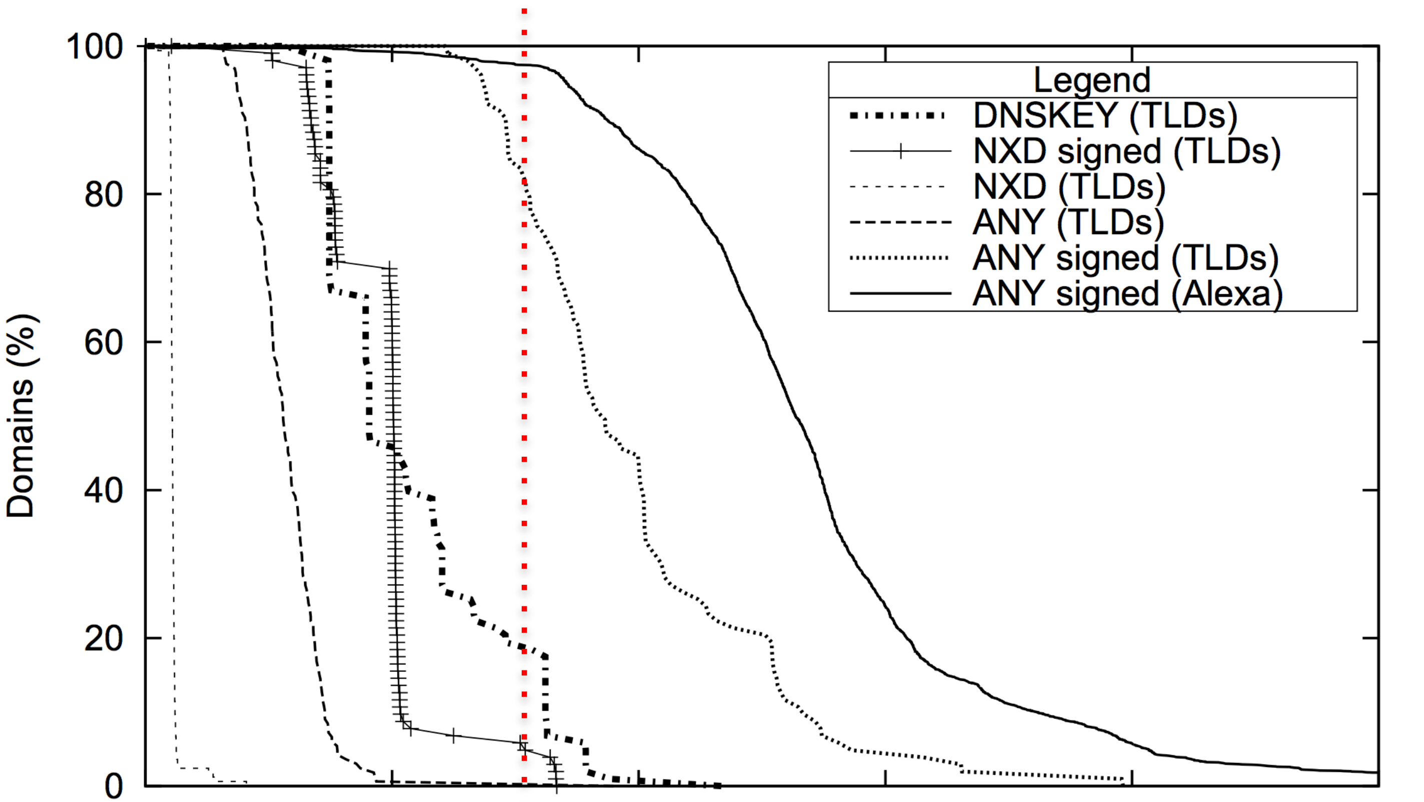
- Clients and servers can specify padding length in messages
- Method of padding selection is left *unspecified*

[1] https://tools.ietf.org/html/rfc7830

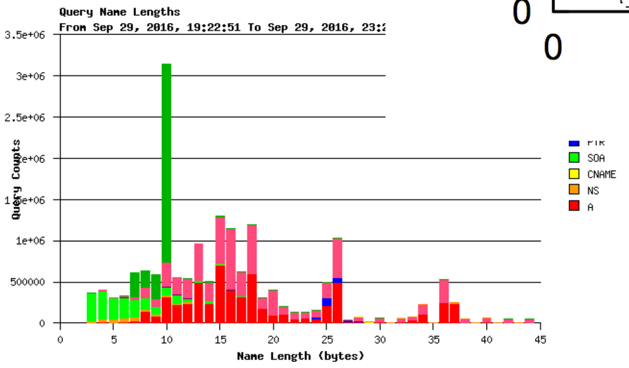
Maximum QNAME Size



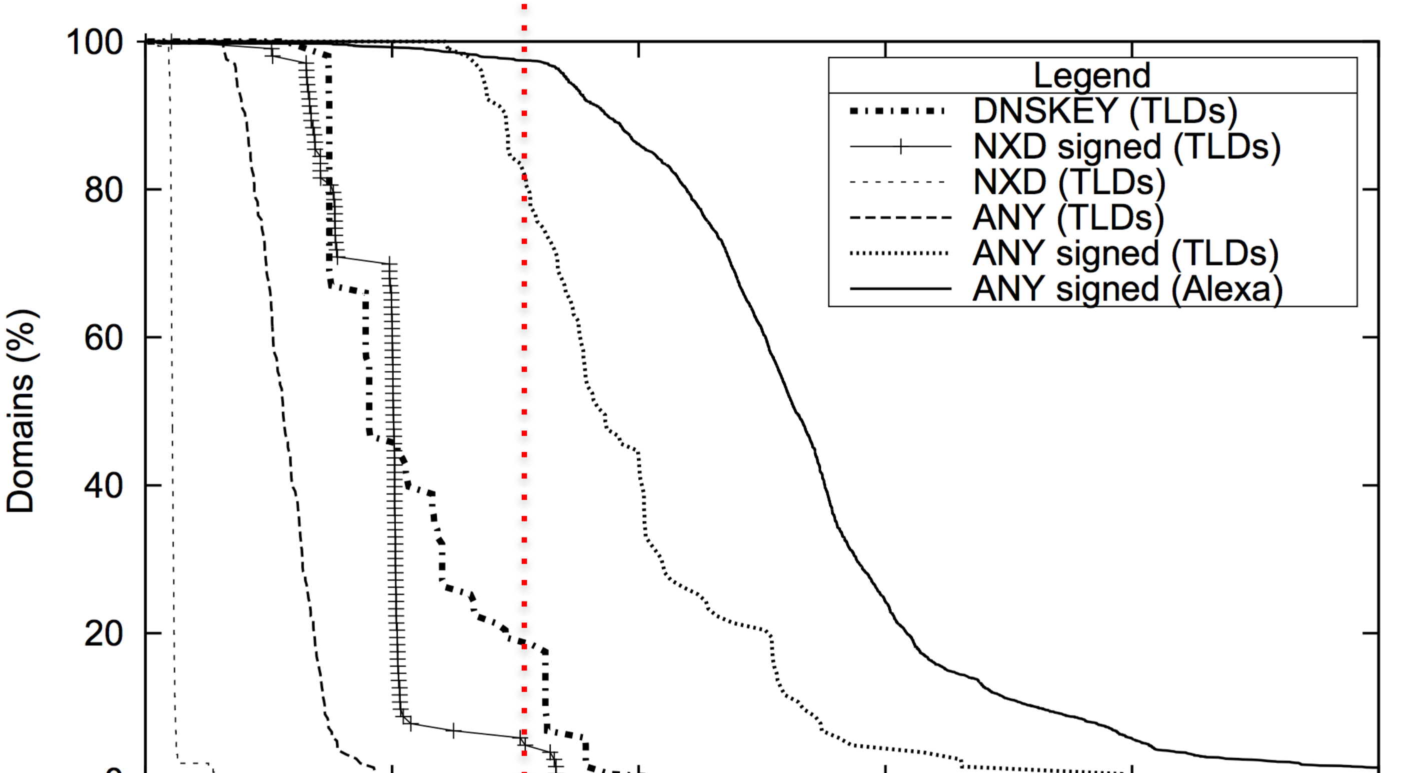
Maximum Response Size

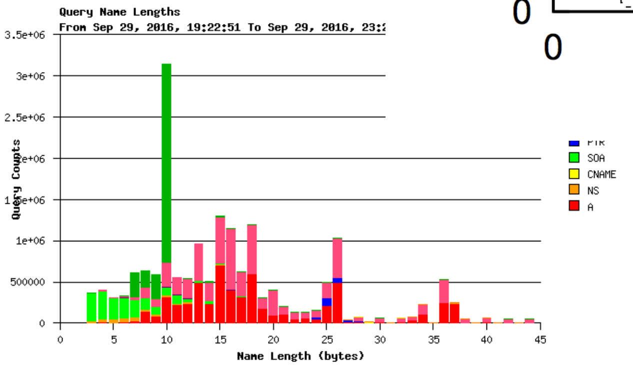


Standard MTU



Maximum Response Size





Standard MTU

Where's the end?

Padding Choices

- Ideally: padding is *uniform*
- Tradeoff: break responses into ”sized tiers”
  - Size $\in$ [1,100] => Tier 1
  - Size $\in$ [101,200] => Tier 2
  - …
  - Size > X => Tier N

Boundaries

How can tier boundaries be selected such that privacy is increased while overhead is decreased?

- – Fewer tiers => more privacy, more overhead
- – More tiers => less privacy, less overhead

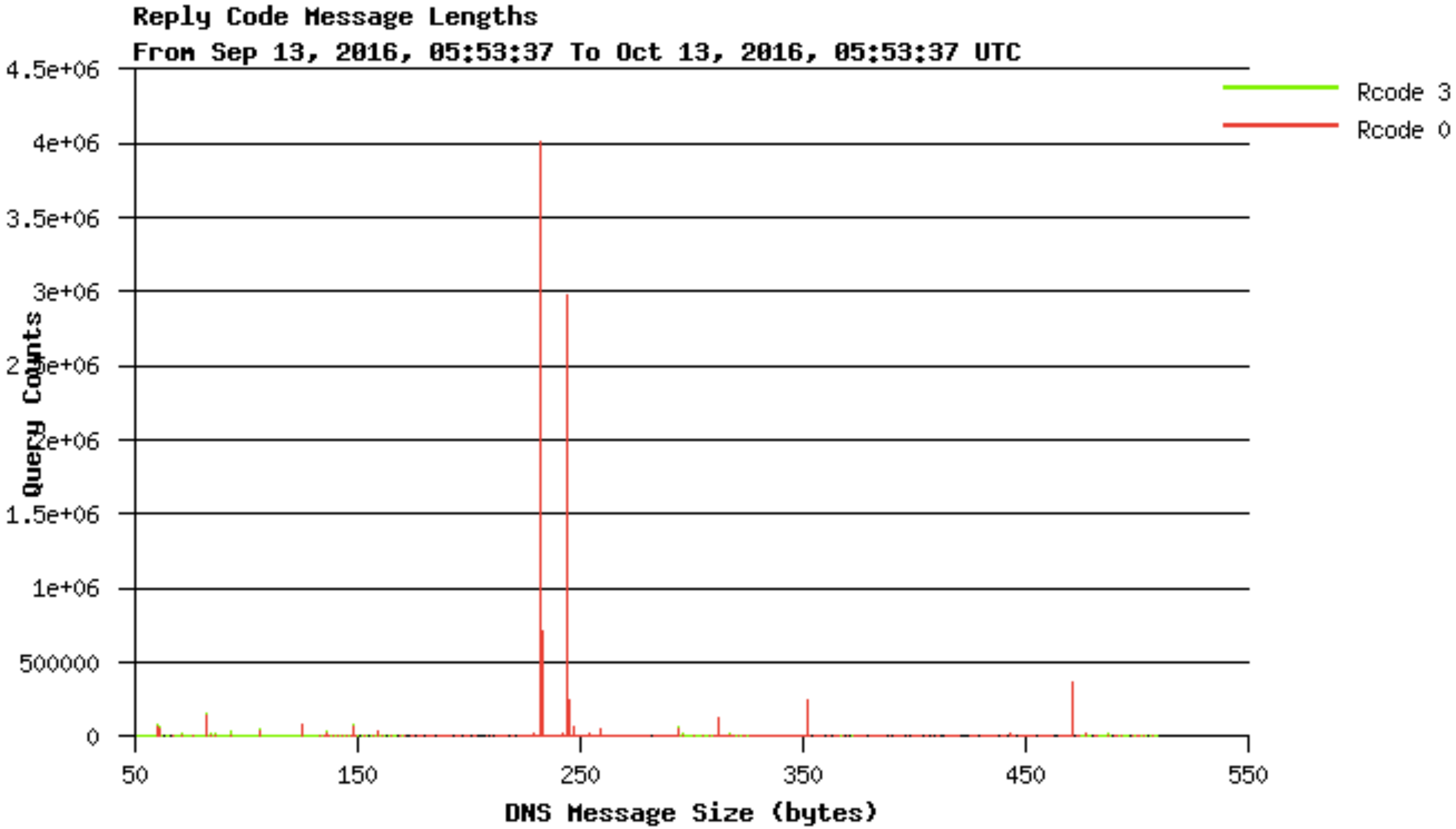
Boundaries

How can tier boundaries be selected such that privacy is increased while overhead is decreased?

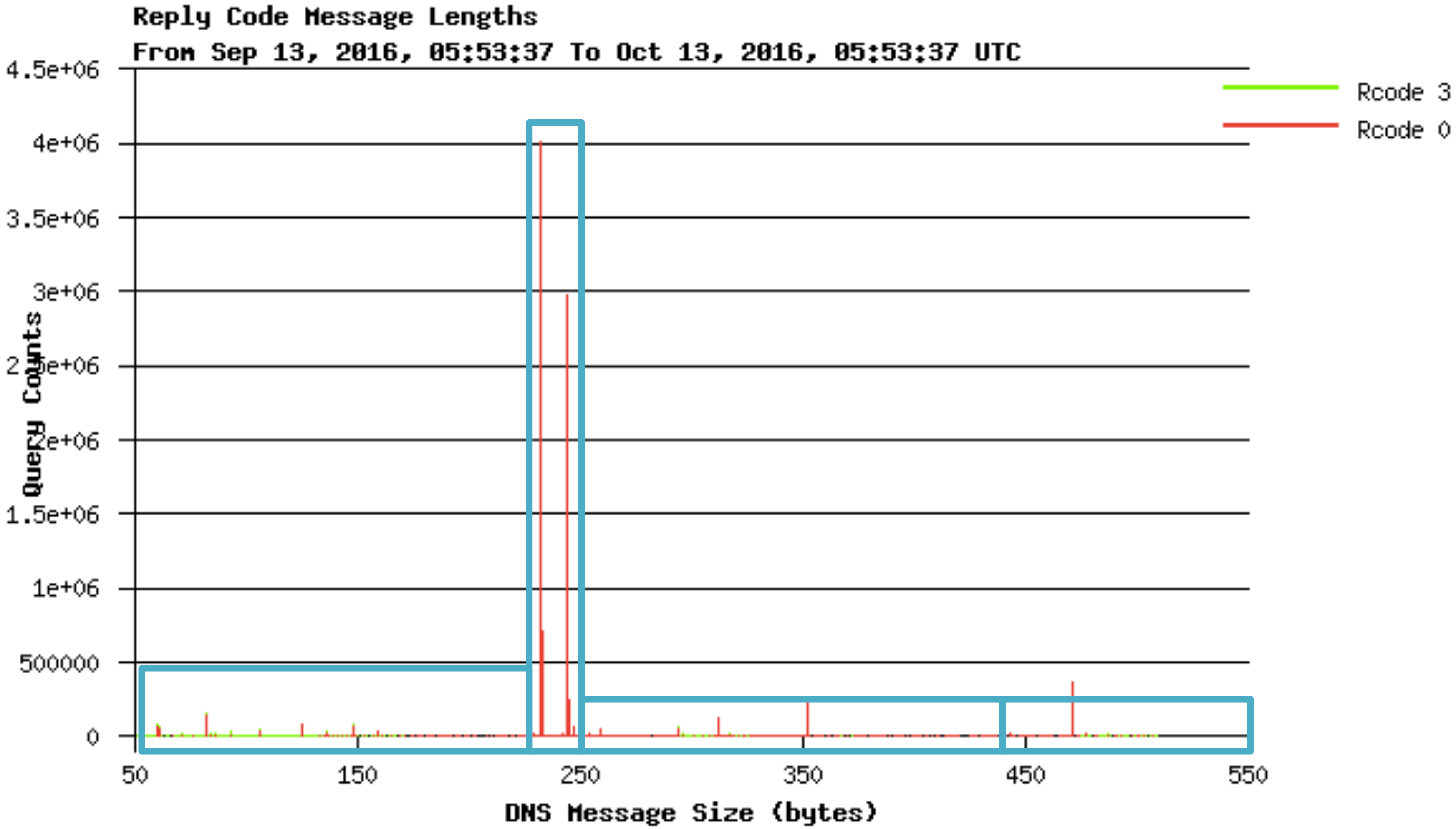
- Fewer tiers => more privacy, more overhead
- More tiers => less privacy, less overhead

Build tiers dynamically based on (cumulative) distribution of requests

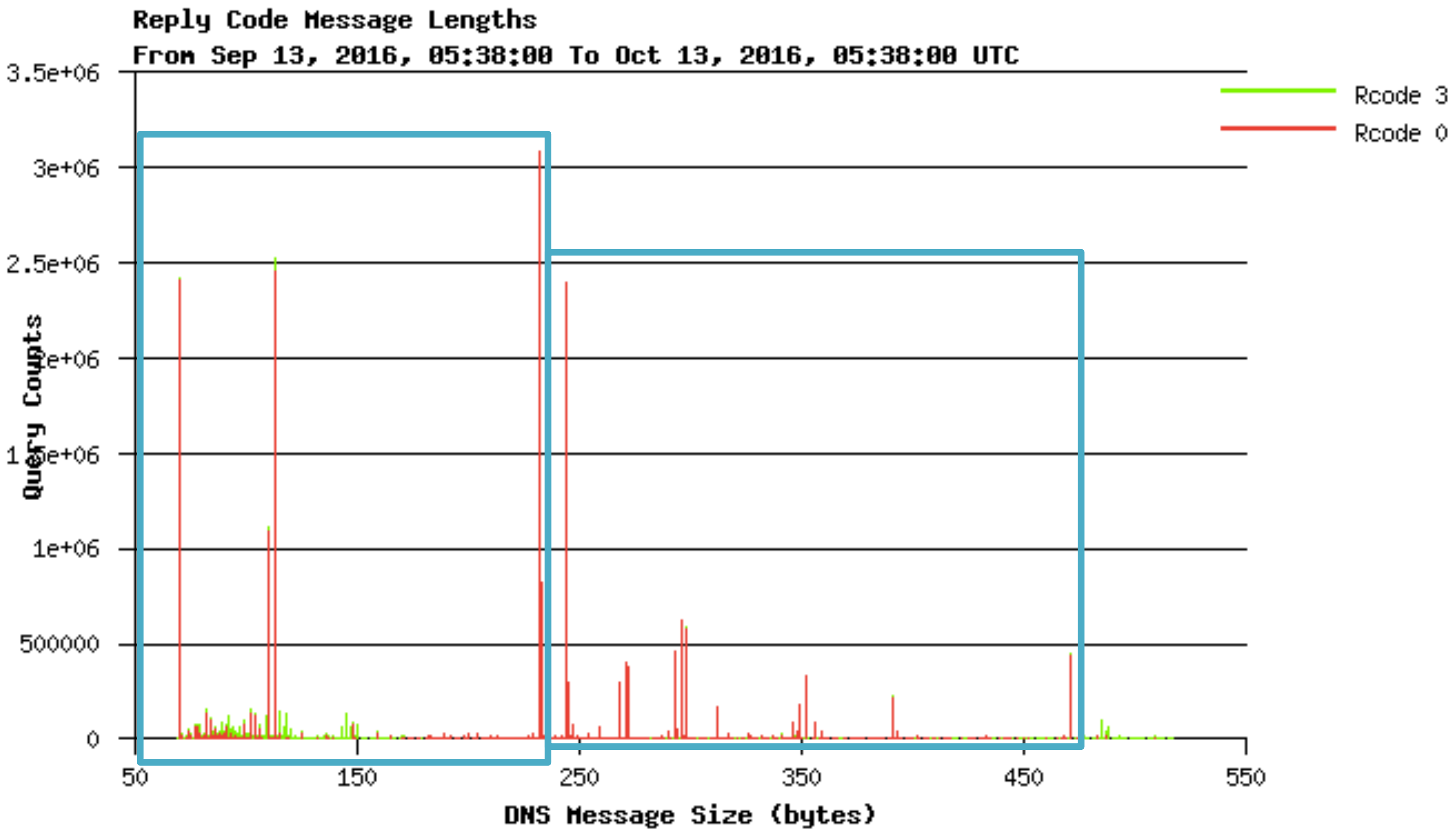
Padding Tiers



Padding Tiers



Padding Tiers

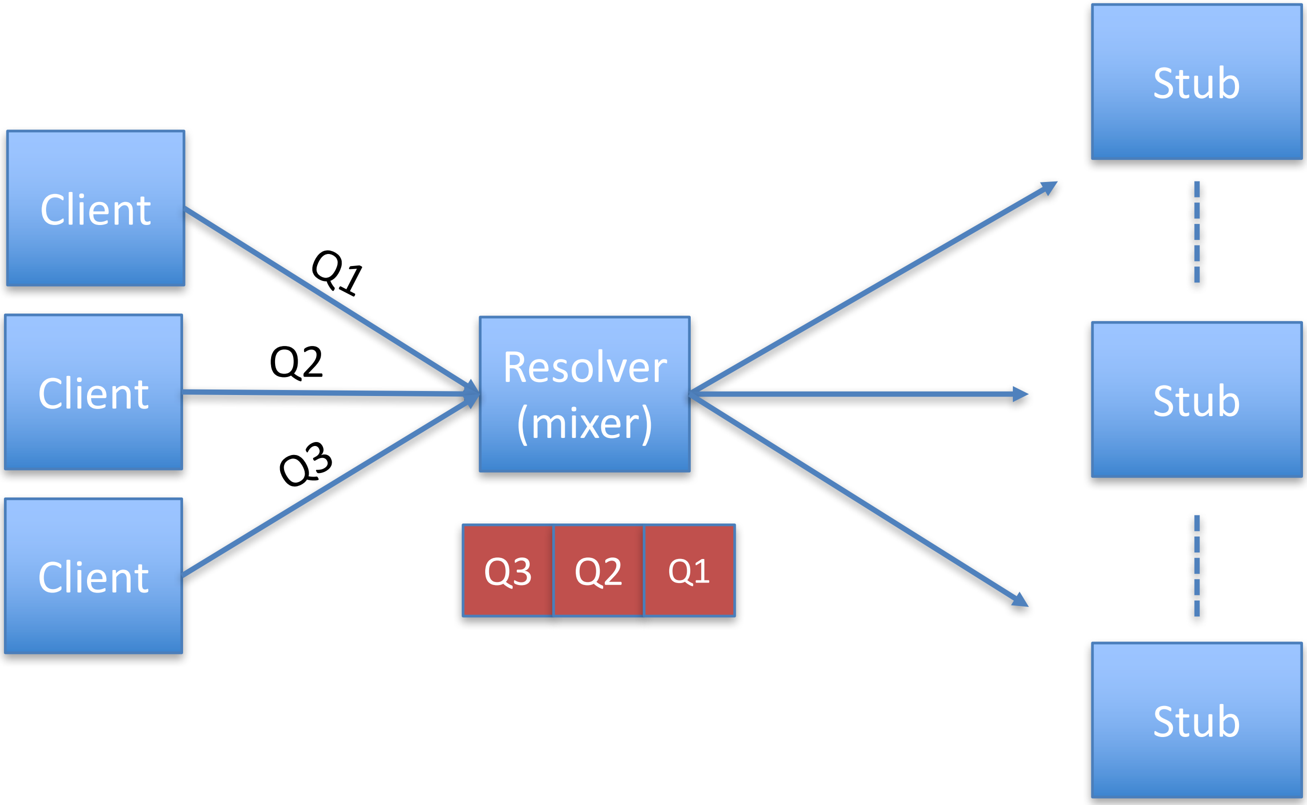


Message Interleaving [frequency, time, chains]

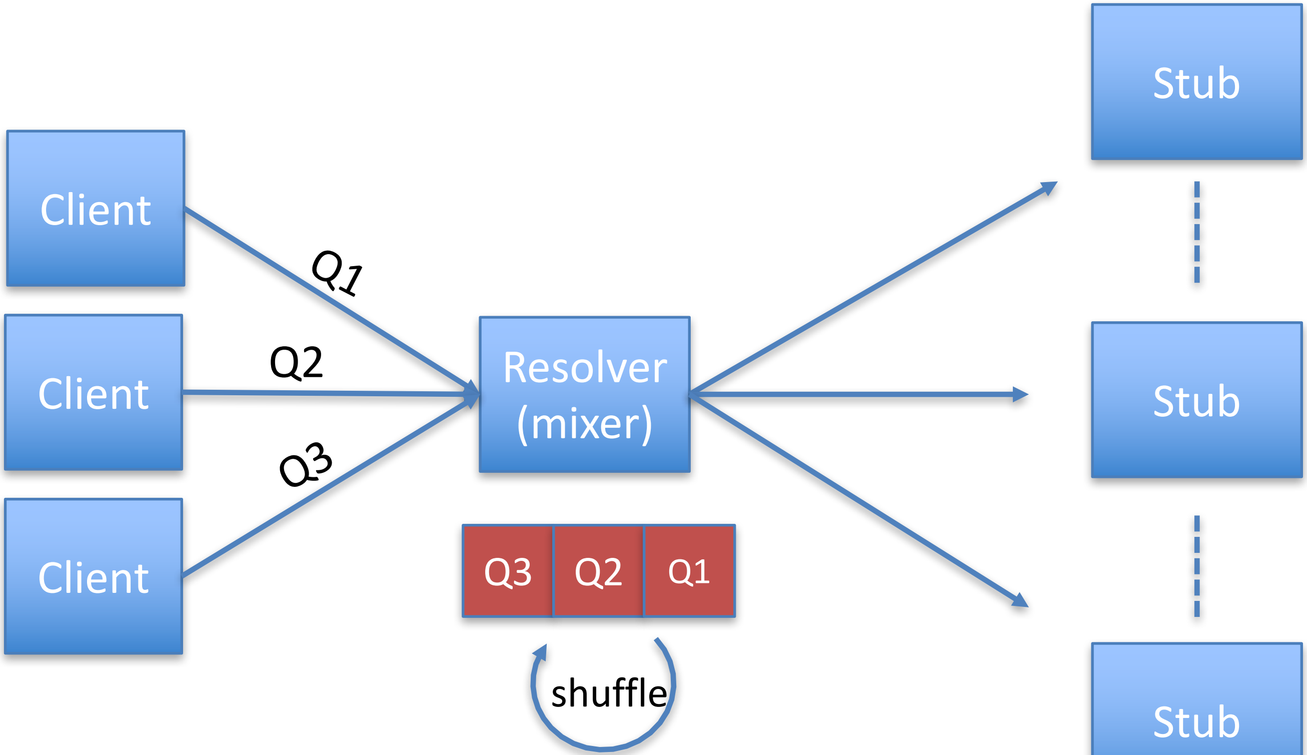
- Requirements: mask query order by interleaving messages
  - Cannot interleave unless we batch queries
  - Want to minimize query delays while maximizing interleaving
- Approach:
  - Batch for RTT* seconds
  - Shuffle packets (queries and responses), send in sequence, repeat

*Average RTT to resolve a single query

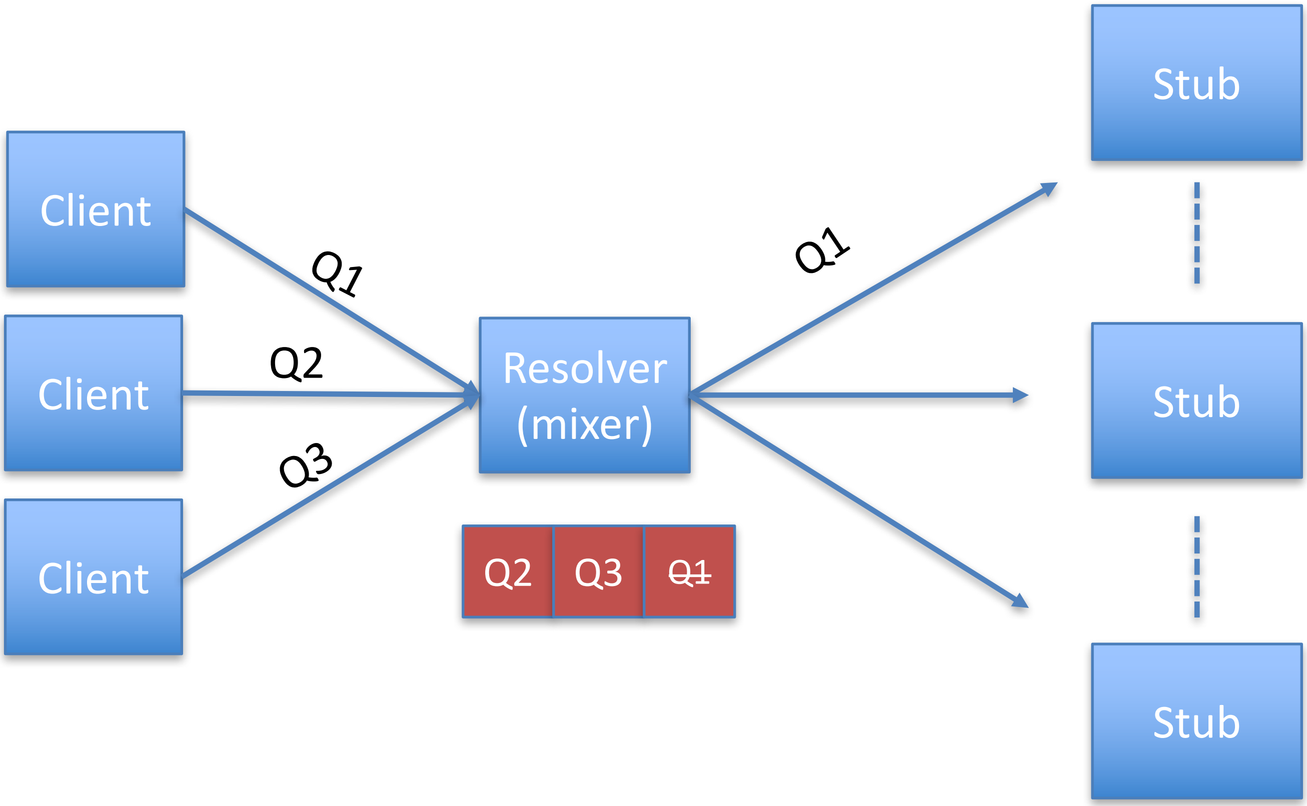
Message Interleaving Overview



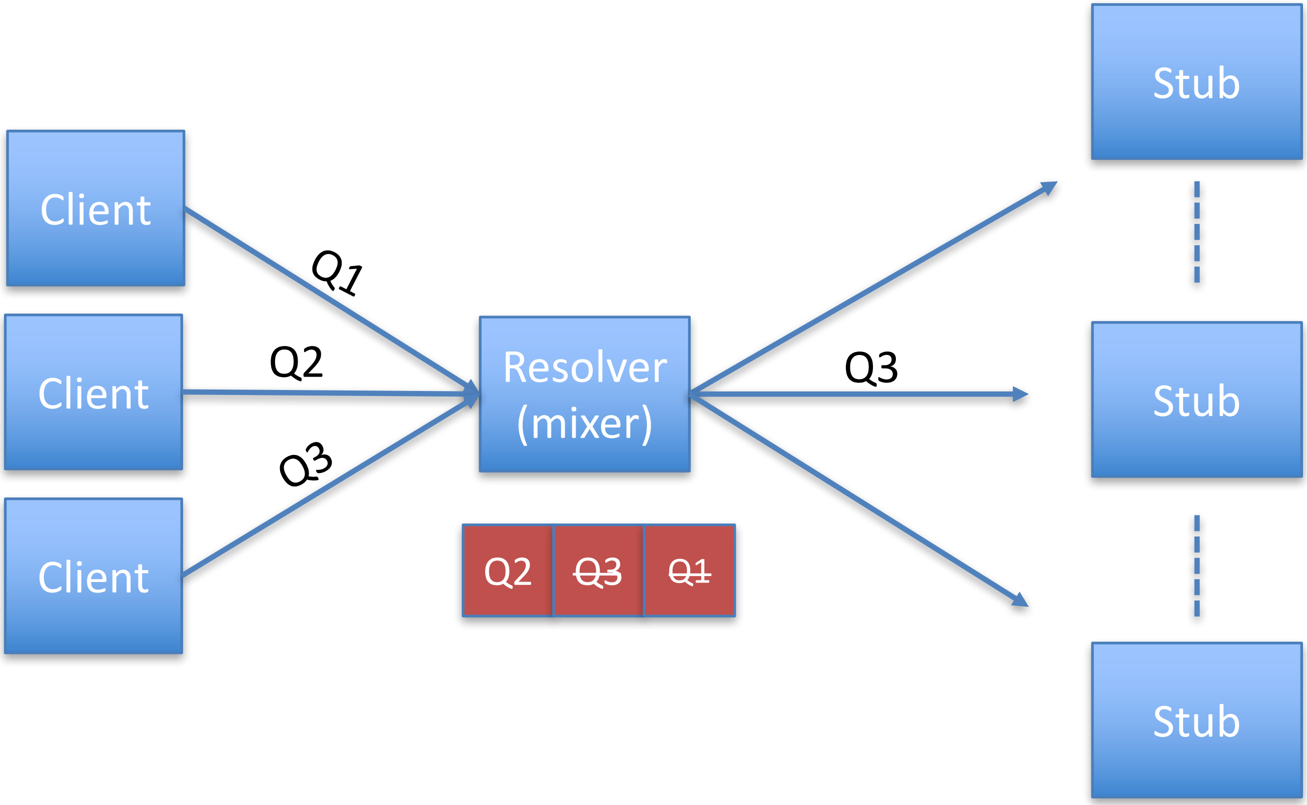
Message Interleaving Overview



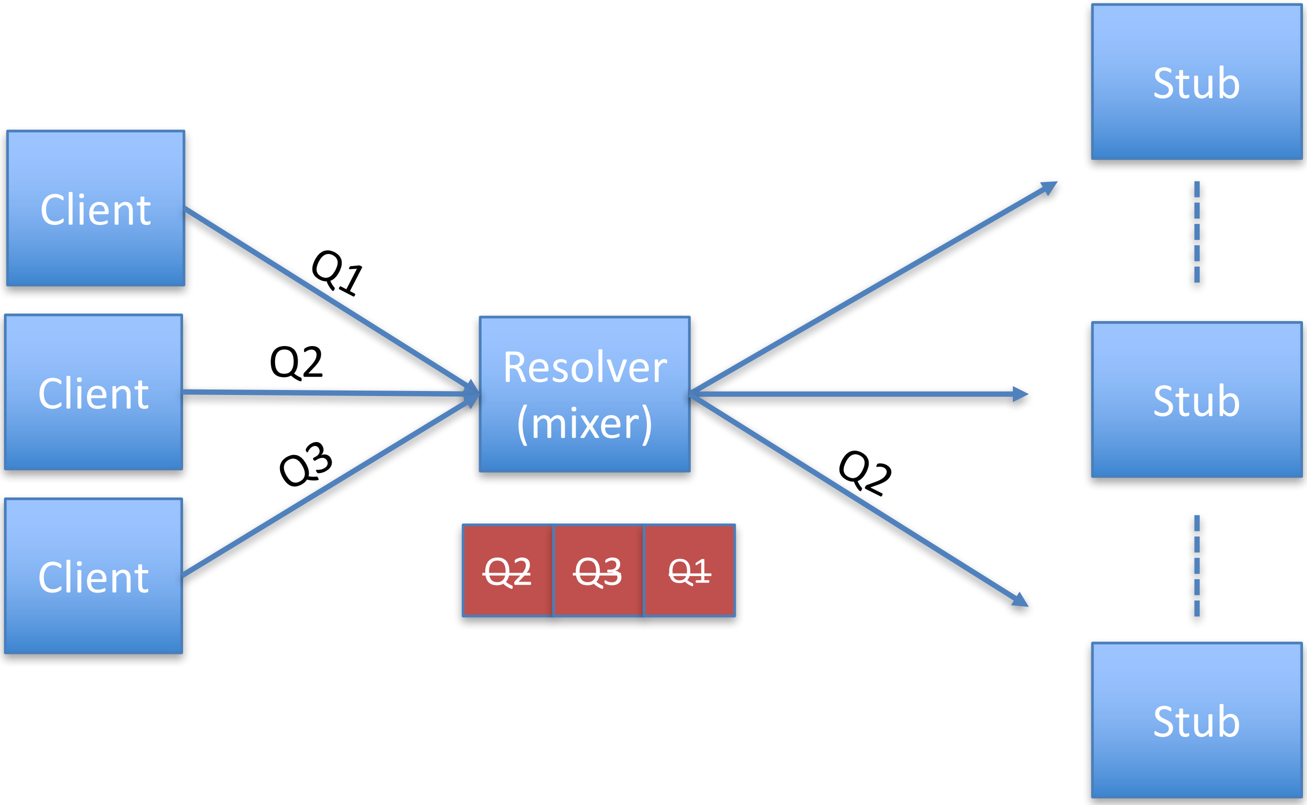
Message Interleaving Overview



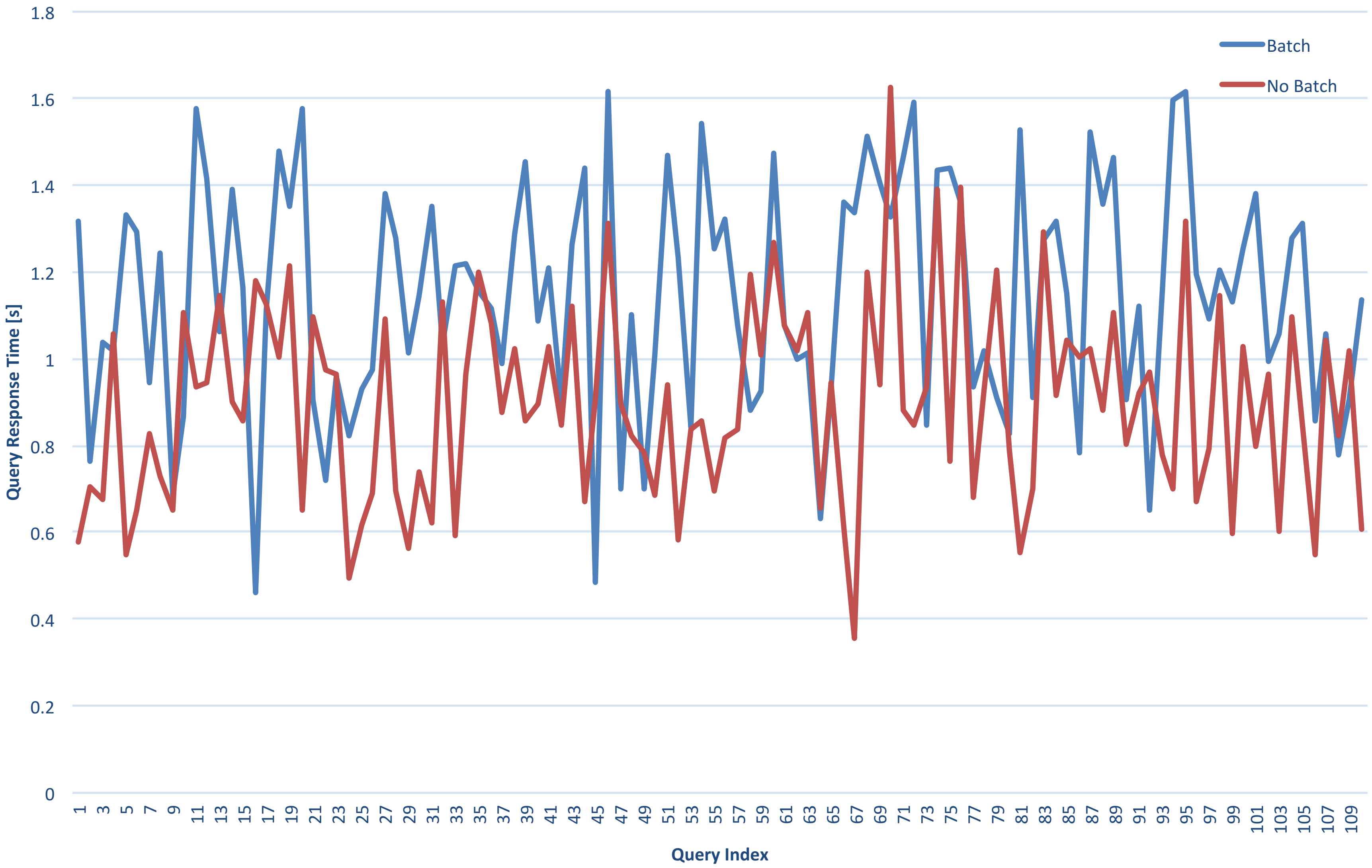
Message Interleaving Overview



Message Interleaving Overview

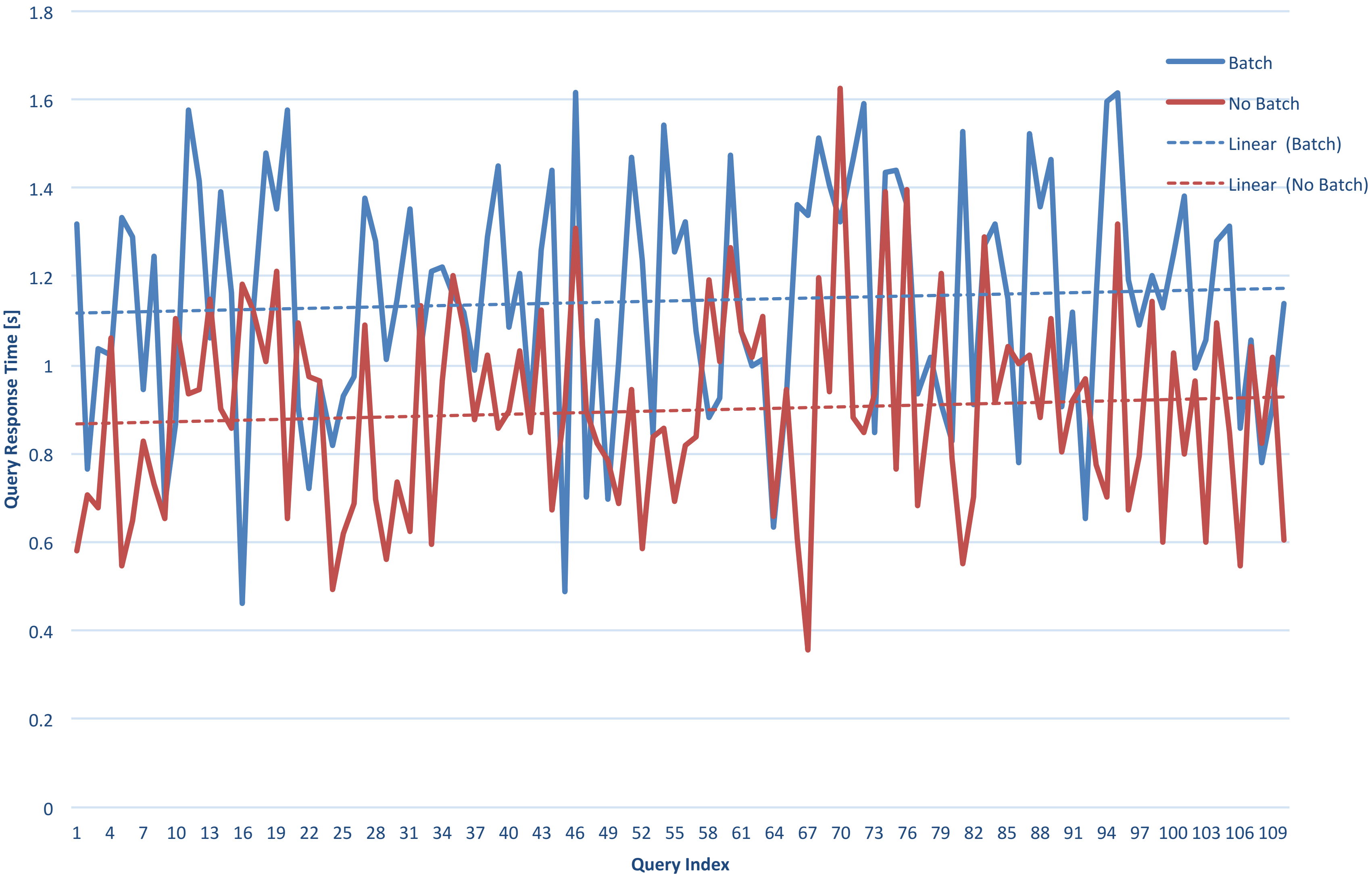


Results



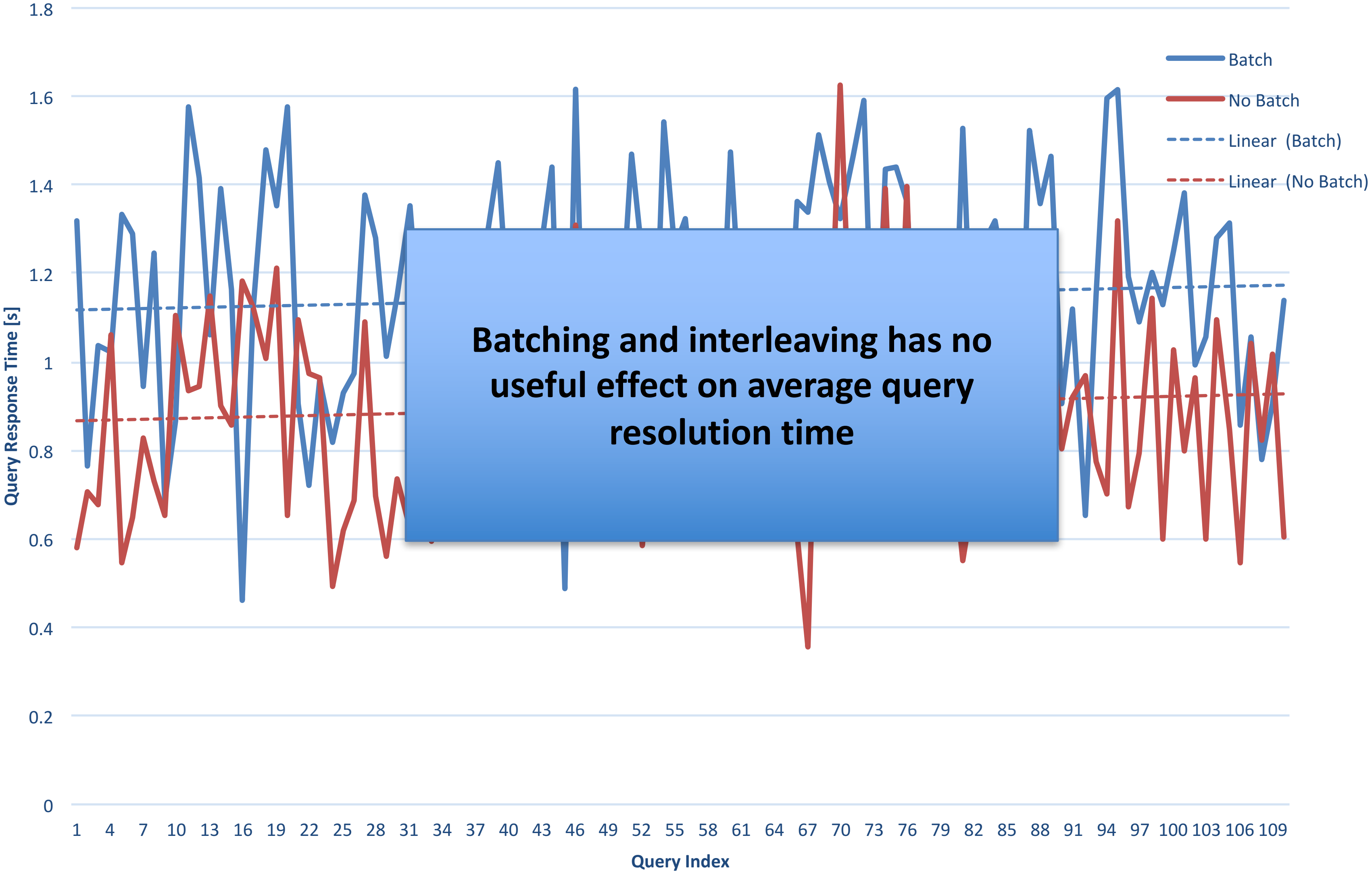
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
Query Response Time [s]
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41
43
45
47
49
51
53
55
57
59
61
63
65
67
69
71
73
75
77
79
81
83
85
87
89
91
93
95
97
99
101
103
105
107
109
Query Index
Batch
No Batch

Results



1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
Query Response Time [s]
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94 97 100 103 106 109
Query Index
Batch
No Batch
Linear (Batch)
Linear (No Batch)

Results

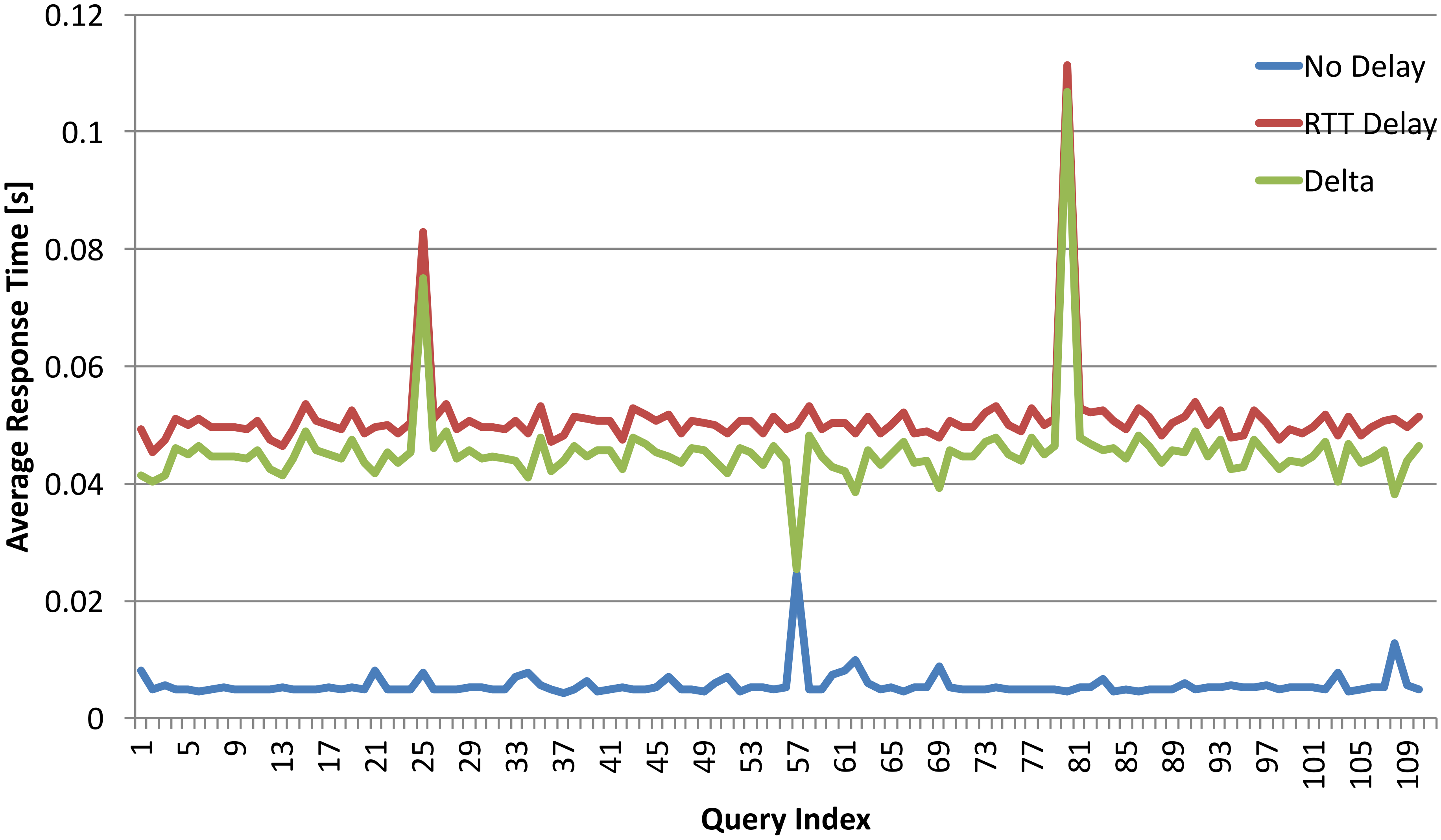


Batching and interleaving has no useful effect on average query resolution time

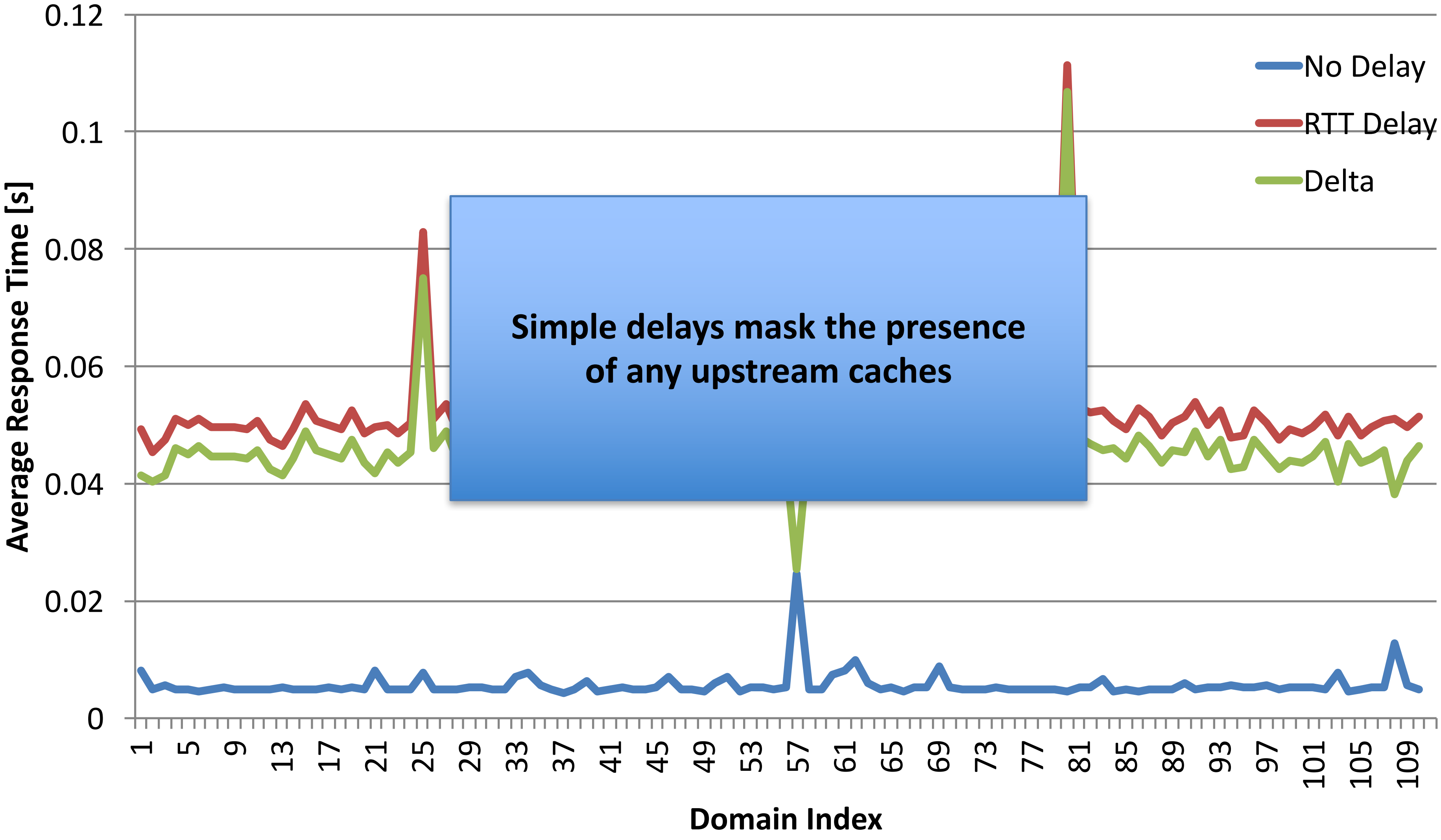
Artificial Resolver Delays [time]

- Requirements: introduce artificial delays in resolvers to mask timing side channels (even with RANSes)
- Approach:
  - If data not cached, resolve the request and record the RTT
  - Else, wait for the previously recorded RTT before returning the response

Delay Effects



Delay Effects



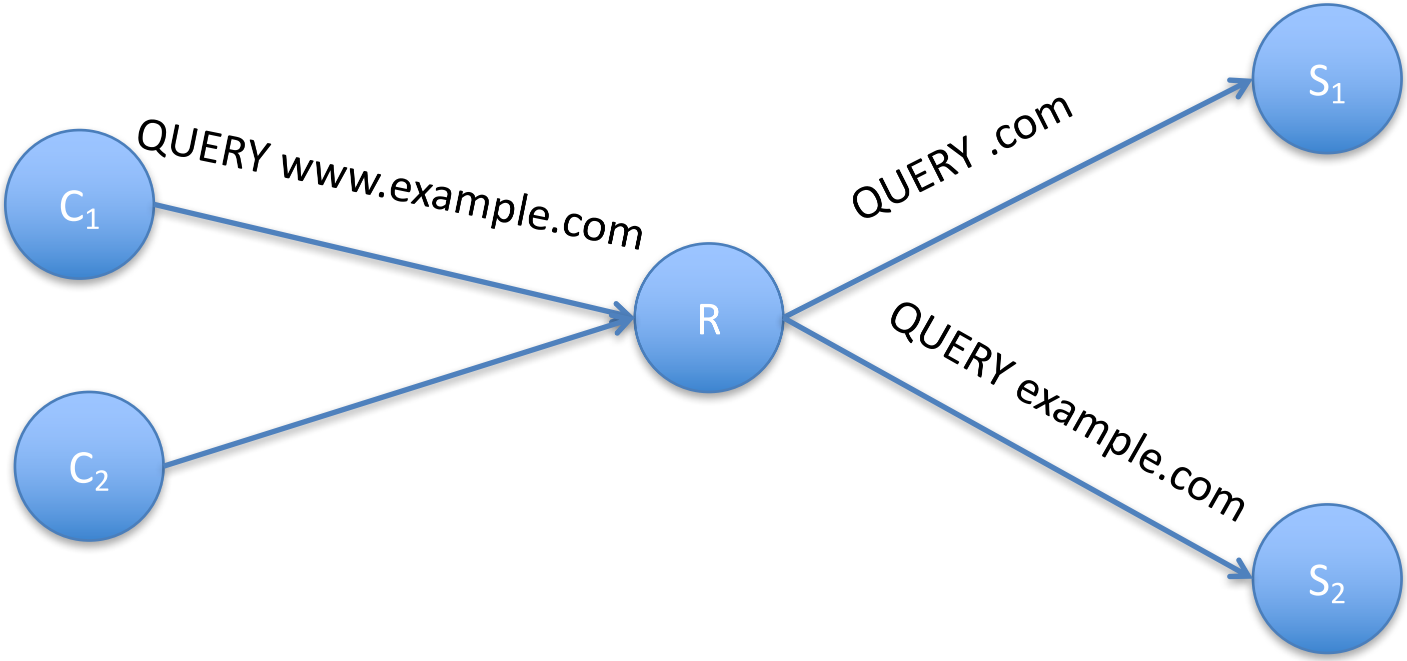
Simple delays mask the presence of any upstream caches

Side Effects and Questions

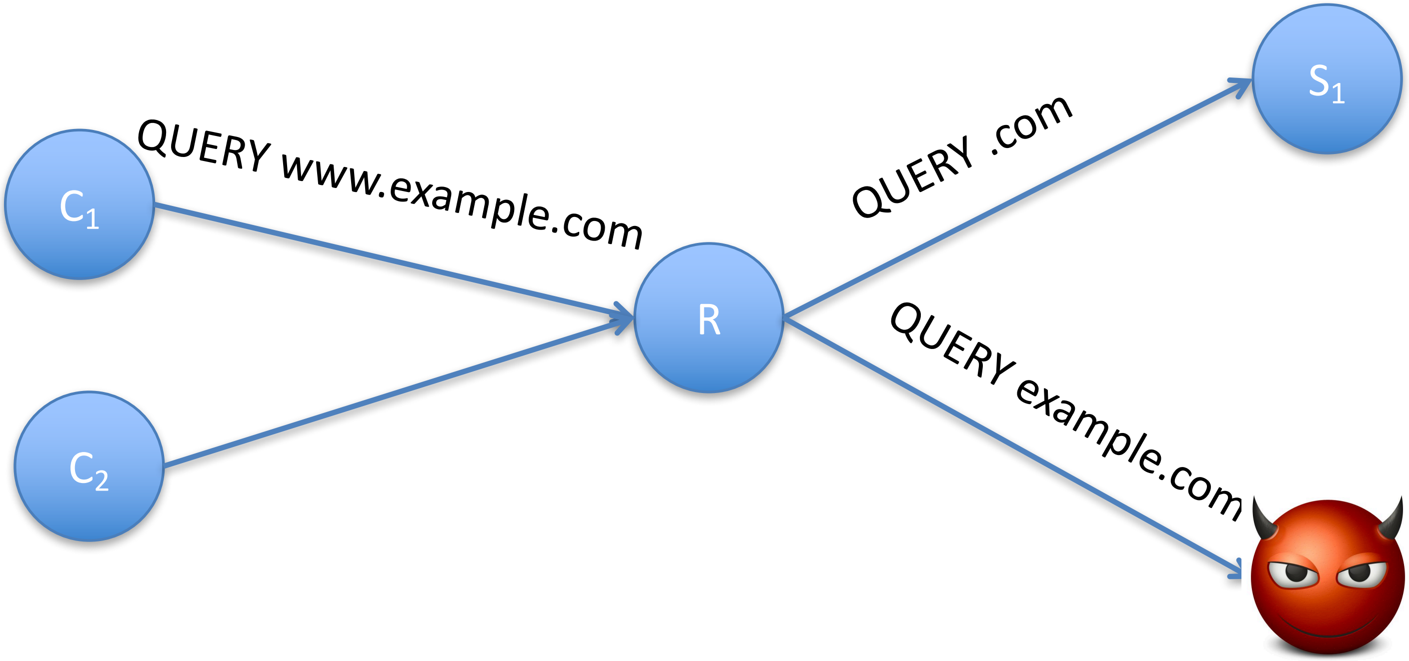
- Worst-case latency for clients
  - *Is < 0.1s noticeable?*
- Per-record query delays can reveal information about different resolution strategies
  - *Should the delay **always** be the worst case across all records?*

ANONYMITY

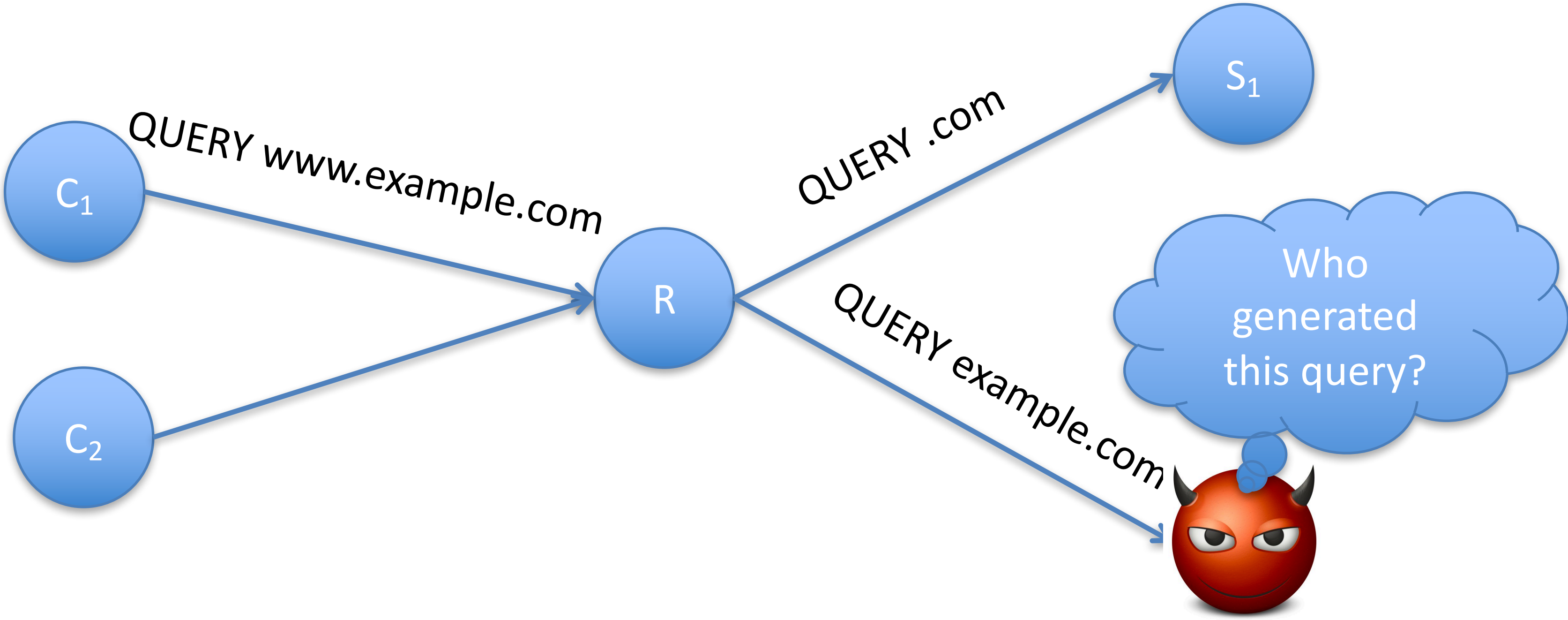
Adversarial Model



Adversarial Model



Adversarial Model



De-Anonymizing Attack

- Goal: use information in queries to link them to specific clients

- Many features to choose from:
  - Query length
  - Query target name
  - Query frequency (windowed)
  - Query single component differences
  - Query entropy
  - Query target address
  - …

- Other possibilities:
  - Resolution chain (not visible to stub adversary)

Approach

Data

- Capture DNS packet traces for small set of users over a single day for numerous days
- One day becomes training data, the rest is test data

Computation

```
for classifier in classifiers:
    for feature_set in combinations(features):
        classifier.train(feature_set, training_data)
        error_rate = classifier.process(feature_set, live_data)
```

Classifiers

We sampled a number of classifiers:

- SVM
- Linear classifier (logistic regression)
- SGD (stochastic gradient descent)
- Decision Tree

Results*

Classifier	Feature(s)	Error Rate
Linear	Query length	0.5185
SVM	Query length	0.5076
SGD	Query length	0.5077
Linear	Query length, query frequency	0.6042
SVM	Query length, query frequency	0.5895
SGD	Query length, query frequency	0.5425
Linear	Query length, query frequency, query target name	0.5293
SVM	Query length, query frequency, query target name	0.5224
SGD	Query length, query frequency, query target name	0.5342

*subset of the entire result set

Results*

Classifier	Feature(s)	Error Rate
Linear	Query length	0.5185
SVM	Query length	0.5076
SGD	Query lengt	0.5077
Linear	Query lengt	0.6042
SVM	Query lengt	0.5895
SGD	Query lengt	0.5425
Linear	Query lengt	0.5293
SVM	Query length, query frequency, query target name	0.5224
SGD	Query length, query frequency, query target name	0.5342

None of these features are helpful (and that's good)

*subset of the entire result set

Query Chaffing

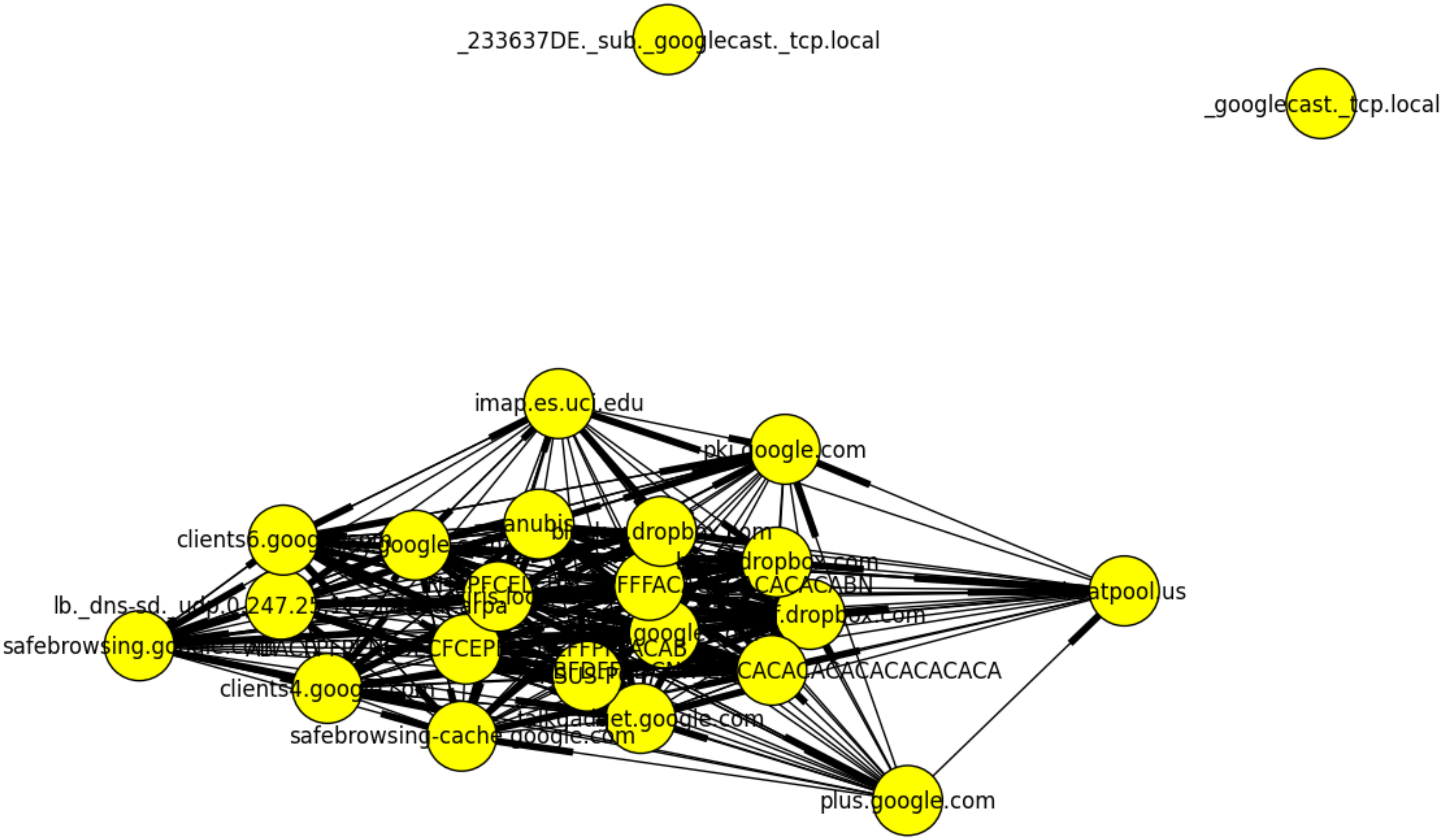
- Requirements:
  - Chaffing should *look similar* to existing queries
  - *Rate* should resemble legitimate traffic
- Idea:
  - Using DNS packet traces, build a weighted directed graph of domain relationships
  - Sample chaff traffic from neighbors of past queries

Domain Graphs

G = (V,E) such that

- V is the set of domains (QNAMEs)
- *$(u \in V, v \in V) \in E$* iff *v* is queried **after** *u* from the same address
  - Implies that there is some relationship between the two domains
  - twitter -> facebook -> youtube

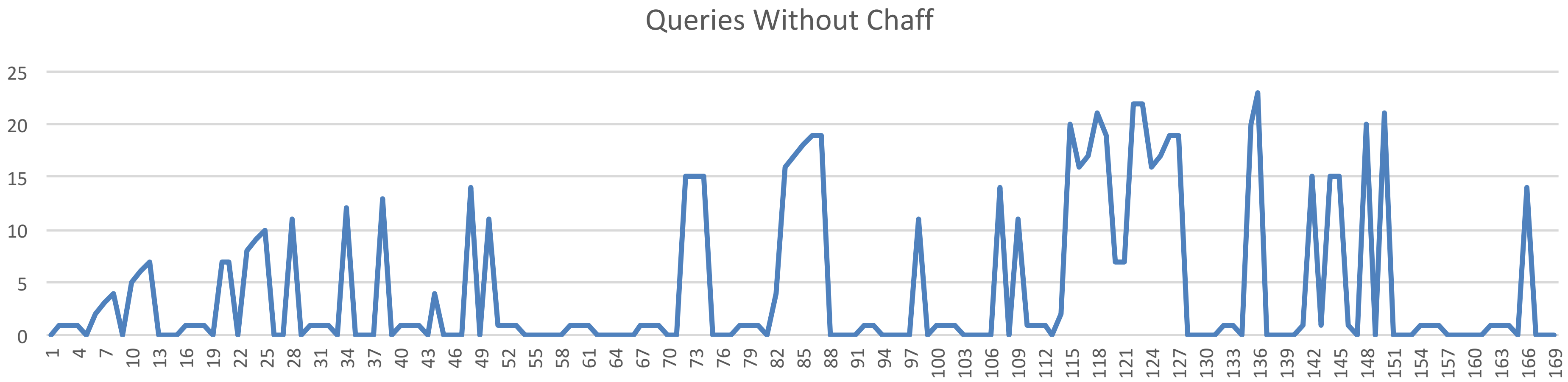
Example

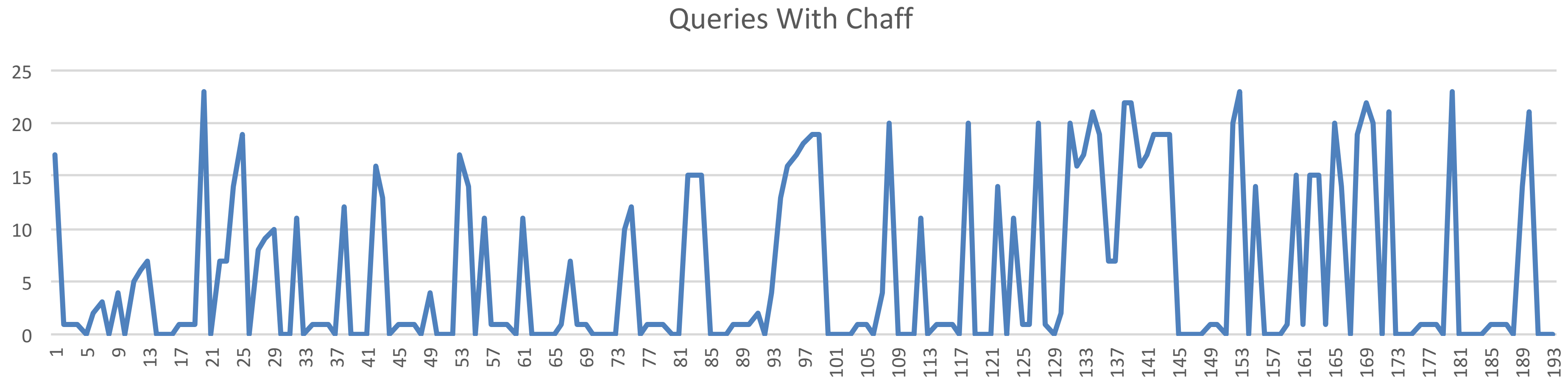

_233637DE._sub._googlecast._tcp.local
_googlecast._tcp.local
imap.es.uci.edu
pki.google.com
anubis
dropbox.com
clients6.google
google
dropbox.com
FFFAC
CACACACABN
atpool.us
lb._dns-sd._udp.0.247.25
arpa
.dropbox.com
google
safebrowsing.go
CFCEP
ACAB
CACACACACACACACACA
clients4.goog
talkgadget.google.com
safebrowsing-cache.google.com
plus.google.com

Approach

- Perform random traversal of the domain graph
- Advance at the average query rate

Results


Queries Without Chaff


Queries With Chaff

Wrapping Up

- Examined privacy-enhancing mechanisms have marginal (if any) benefits
  - **Artificial cache delays**: only measure that seems to truly help while being minimally intrusive
- Anonymity (against the limited adversary) seems safe
  - Stronger adversaries (closer to the clients) will have an easier time
  - Query chaffing helps unify traffic patterns but at significant cost

QUESTIONS?

FIRE AWAY!

Special thanks to Verisign for their support of this work!