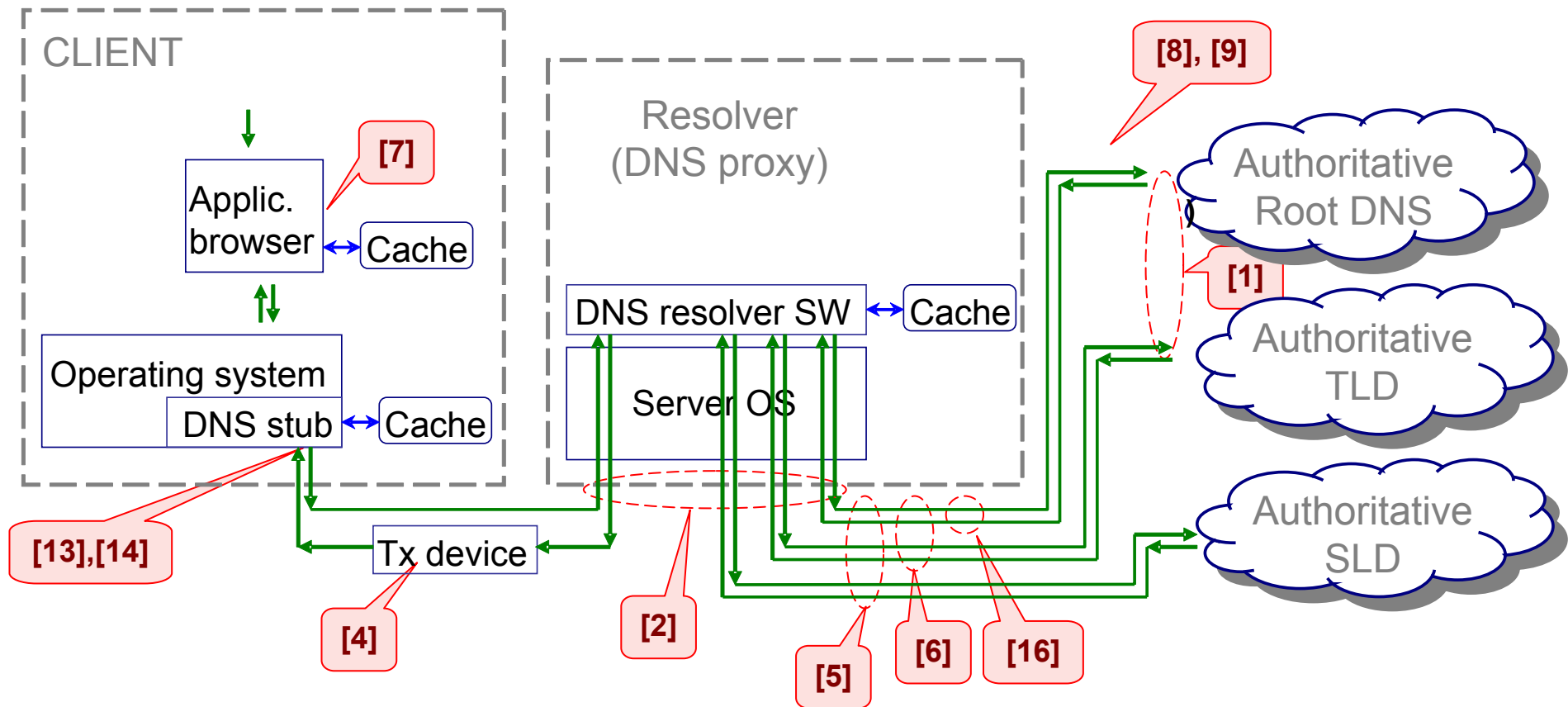


# DNS(SEC) client analysis



# 'Overview' DNS traffic analysis



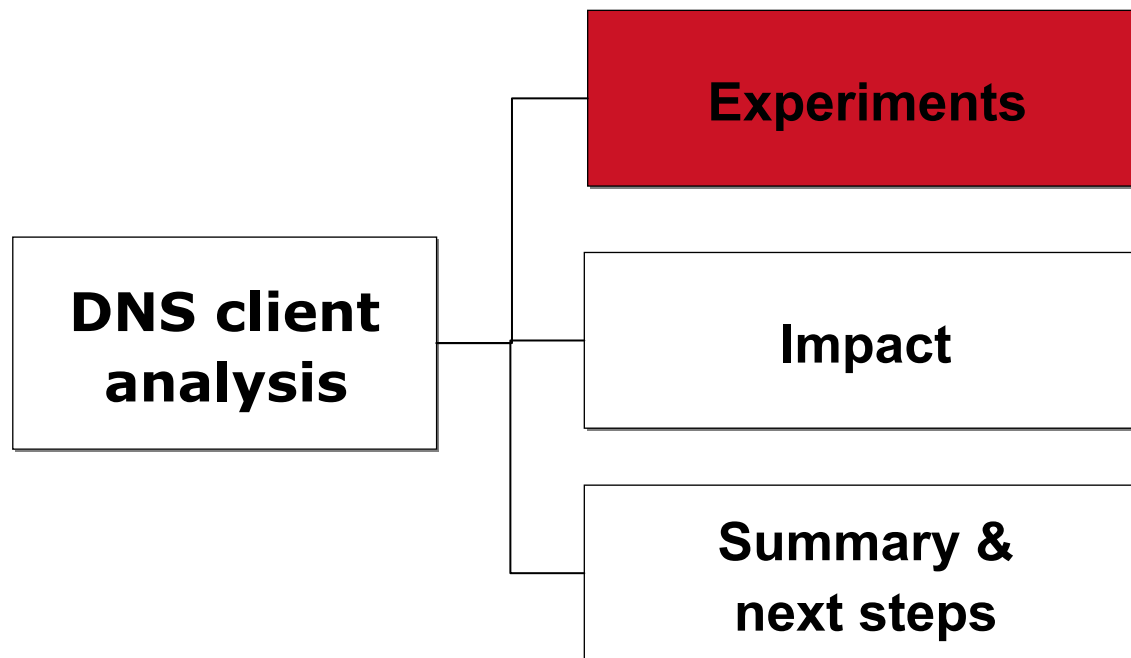
- › Focus of DNS analysis has been on resolver and authoritative ⇔ bulk data analysis

## Key question:

- › How will DNSSEC change the behavior of DNS client querying?

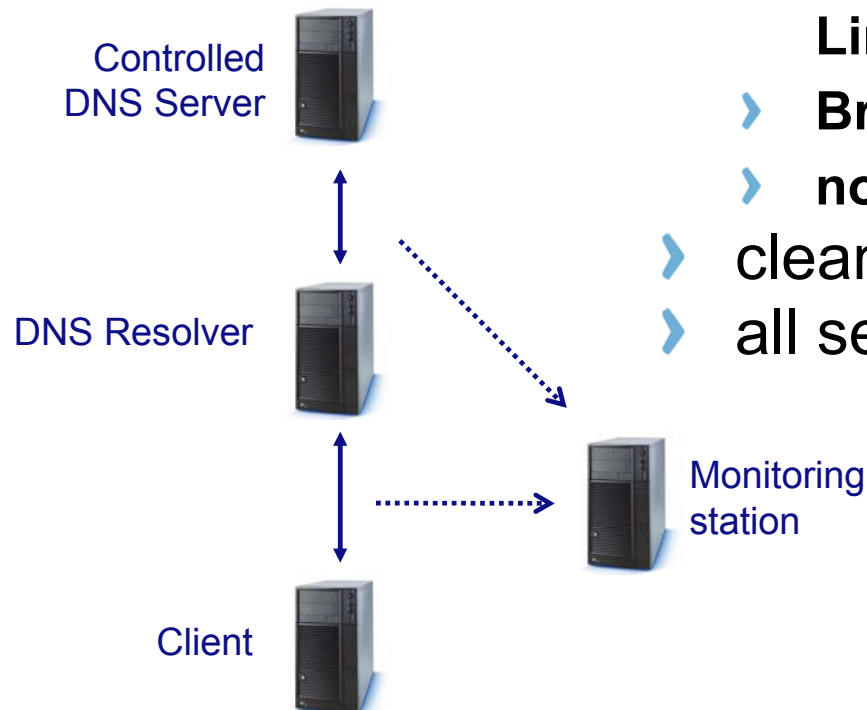
More specific ...

- › How do DNS stub resolvers react to response types such as ServFail, responses > 512 Bytes, ...?



## Experimental set-up

- › Configure OS / browser on client machine
  - › OS: Windows XP, Windows 7, Ubuntu Linux, Mac OSX
  - › Browsers: IE, Firefox, Chrome, Safari
  - › not all combi's, but quite some ...
- › clean OS image
- › all settings left on defaults



## Test execution

- Execute test run
  - query each URLs with predefined response (Idns tool)
    - **Valid, Valid (>512 Bytes), NXdomain, Partial, ServFail, No reply, Truncated, Recursion refused**
  - query via ping (=> OS only) and via browser (=> browser & OS)
  - repeat query once to check impact of caching
  
- **Observe the number of repeated queries and delays**



## Example of DNS client behaviour: Linux-Ubuntu /w Firefox

```

15:26:38.694678 IP 10.0.3.2.56600 > 10.0.2.1.53: 7000+ AAAA? servfail.dnslab.nl. (36)
15:26:38.704409 IP 10.0.2.1.53 > 10.0.3.2.56600: 7000 ServFail 0/0/0 (36)
15:26:38.704779 IP 10.0.3.2.46832 > 10.0.2.1.53: 7000+ AAAA? servfail.dnslab.nl. (36)
15:26:38.711533 IP 10.0.2.1.53 > 10.0.3.2.46832: 7000 ServFail 0/0/0 (36)
15:26:38.712139 IP 10.0.3.2.34859 > 10.0.2.1.53: 751+ AAAA? servfail.dnslab.nl. (36)
15:26:38.720254 IP 10.0.2.1.53 > 10.0.3.2.34859: 751 ServFail 0/0/0 (36)
15:26:38.722147 IP 10.0.3.2.60413 > 10.0.2.1.53: 751+ AAAA? servfail.dnslab.nl. (36)
15:26:38.732281 IP 10.0.2.1.53 > 10.0.3.2.60413: 751 ServFail 0/0/0 (36)
15:26:38.732819 IP 10.0.3.2.53267 > 10.0.2.1.53: 62476+ A? servfail.dnslab.nl. (36)
15:26:38.741631 IP 10.0.2.1.53 > 10.0.3.2.53267: 62476 ServFail 0/0/0 (36)
15:26:38.742221 IP 10.0.3.2.55543 > 10.0.2.1.53: 62476+ A? servfail.dnslab.nl. (36)
15:26:38.750843 IP 10.0.2.1.53 > 10.0.3.2.55543: 62476 ServFail 0/0/0 (36)
15:26:38.750843 IP 10.0.3.2.55146 > 10.0.2.1.53: 40336+ A? servfail.dnslab.nl. (36)
15:26:38.757800 IP 10.0.2.1.53 > 10.0.3.2.55146: 40336 ServFail 0/0/0 (36)
15:26:38.758084 IP 10.0.3.2.55095 > 10.0.2.1.53: 40336+ A? servfail.dnslab.nl. (36)
15:26:38.768255 IP 10.0.2.1.53 > 10.0.3.2.55095: 40336 ServFail 0/0/0 (36)
15:26:38.769784 IP 10.0.3.2.33077 > 10.0.2.1.53: 38673+ AAAA? servfail.dnslab.nl. (36)
15:26:38.776757 IP 10.0.2.1.53 > 10.0.3.2.33077: 38673 ServFail 0/0/0 (36)
15:26:38.776971 IP 10.0.3.2.52085 > 10.0.2.1.53: 38673+ AAAA? servfail.dnslab.nl. (36)
15:26:38.787268 IP 10.0.2.1.53 > 10.0.3.2.52085: 38673 ServFail 0/0/0 (36)
15:26:38.787583 IP 10.0.3.2.60192 > 10.0.2.1.53: 55536+ AAAA? servfail.dnslab.nl. (36)
15:26:38.797645 IP 10.0.2.1.53 > 10.0.3.2.60192: 55536 ServFail 0/0/0 (36)
15:26:38.797985 IP 10.0.3.2.46728 > 10.0.2.1.53: 55536+ AAAA? servfail.dnslab.nl. (36)
15:26:38.803552 IP 10.0.2.1.53 > 10.0.3.2.46728: 55536 ServFail 0/0/0 (36)
15:26:38.803796 IP 10.0.3.2.60114 > 10.0.2.1.53: 40195+ A? servfail.dnslab.nl. (36)
15:26:38.810014 IP 10.0.2.1.53 > 10.0.3.2.60114: 40195 ServFail 0/0/0 (36)
15:26:38.810456 IP 10.0.3.2.35270 > 10.0.2.1.53: 40195+ A? servfail.dnslab.nl. (36)
15:26:38.823206 IP 10.0.2.1.53 > 10.0.3.2.35270: 40195 ServFail 0/0/0 (36)
15:26:38.823551 IP 10.0.3.2.57144 > 10.0.2.1.53: 50408+ A? servfail.dnslab.nl. (36)
15:26:38.829749 IP 10.0.2.1.53 > 10.0.3.2.57144: 50408 ServFail 0/0/0 (36)
15:26:38.829961 IP 10.0.3.2.35860 > 10.0.2.1.53: 50408+ A? servfail.dnslab.nl. (36)
15:26:38.836833 IP 10.0.2.1.53 > 10.0.3.2.35860: 50408 ServFail 0/0/0 (36)
    
```

⇔ example: servfail response

3 immediate retries in  
case of servfail response

and IPv4?

OS sends servfail to FireFox;  
FireFox makes OS retry

16 queries in 0.14 seconds

## Browser & OS DNS query amplification

Response type	Firefox	Linux	Total
Valid	x1	x1	x1
NXdomain / Partial	x2	x2	x4
ServFail / No response / Refused	x2	x4	x8
Truncated	x1	1+TCP	1+TCP

Response type	Safari	Mac OSX	Total
Valid	x1	x1	x1
NXdomain / Partial	x1	x2	x2
ServFail / No response / Refused	x1	x4	x4
Truncated	x1	1+TCP	1+TCP

› **DNS query count in case of:**

- › single authoritative NS; in case of primary and secondary => 2x
- › only IPv4; in case of IPv4 and IPv6 => 2x



## Browser & OS DNS query amplification

Response type	IE	Windows XP	Total
Valid / NXdomain	x1	x1	x1
Partial / ServFail / Refused	x1	x1	x1
No response	x1	x5	x5
Truncated	x1	1+TCP	1+TCP

Response type	Chrome	Windows XP	Total
Valid / NXdomain	x1	x1	x1
Partial / ServFail / Refused	x1	x1	x1
No response	x1	x5	x5
Truncated	x1	1+TCP	1+TCP

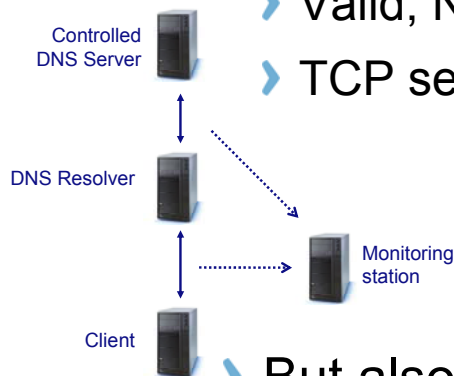
- › In fact, same behaviour for IE, Chrome, Firefox, Safari on Windows XP or Windows 7

## **Other sources of aggressive DNS clients (not investigated)**

- › Greedy – synchronisation apps: bonjour, facebook apps, ...
  - › may generate continuous stream of DNS requests
  
- › Browser pre-fetching
  - › Firefox by default queries “anticipated next URLs” for a page
  - › Chrome pre-fetches stored, successfully retrieved URLs, when started
  
- › Ubuntu Linux: by default no DNS caching

## Impact of the caching resolver

- › Some damping of aggressive client behaviour by (BIND9) resolver
  - › In case of no-response the resolver retries (7 retries, with exponential timer back-off), while holding back client side retries
  - › Valid, NXdomain and truncated responses are cached
  - › TCP session for truncated responses is handled by resolver

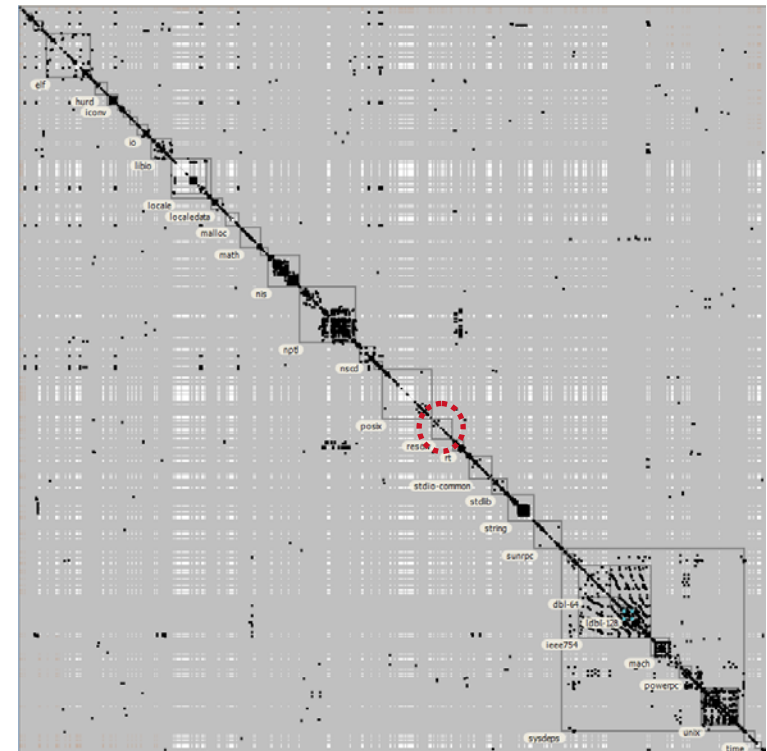


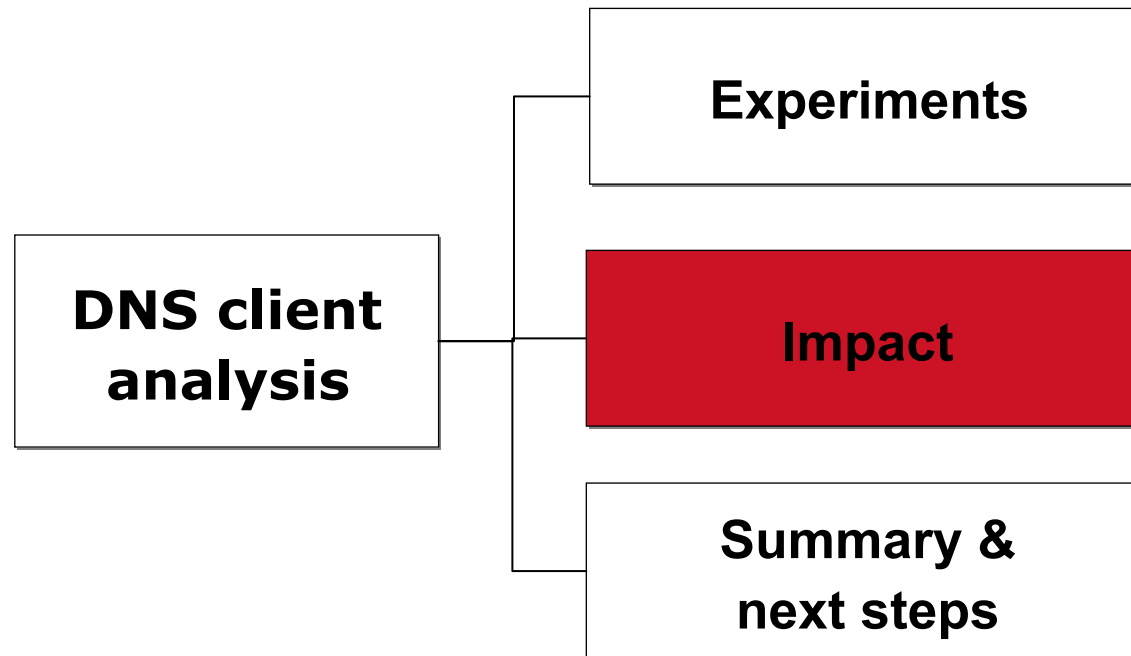
- › But also some amplification / modification by the resolver
  - › Resolver 'double checks' ServFail responses
  - › Unvalidatable response is returned as ServFail to client by non-DNSSEC enabled resolver
  - › Also: partial, recursion refused and timeout are fed back as ServFail

## Causes of aggressive DNS client behavior?

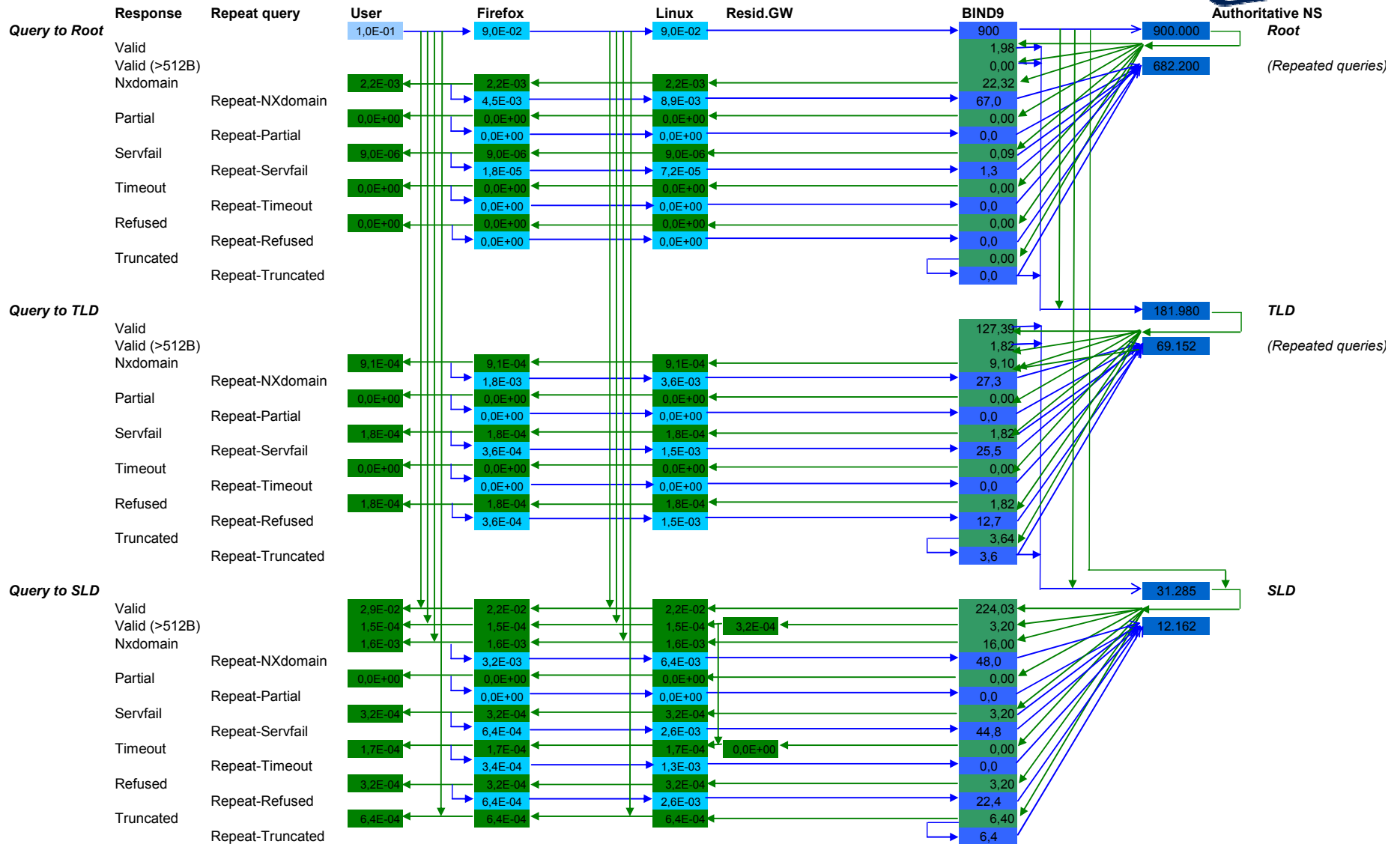


- › GNU Library C ('glibc') DNS service
  - › static code analysis:
    - › overall glibc no ordinary characteristics found
  - › dynamic code analysis of DNS part:
    - › 'responsible' code part is pinpointed
    - › code part is complex ⇔ improvement not found yet
  
- › Ok, before we drill down to the cause ... what's the **impact?**

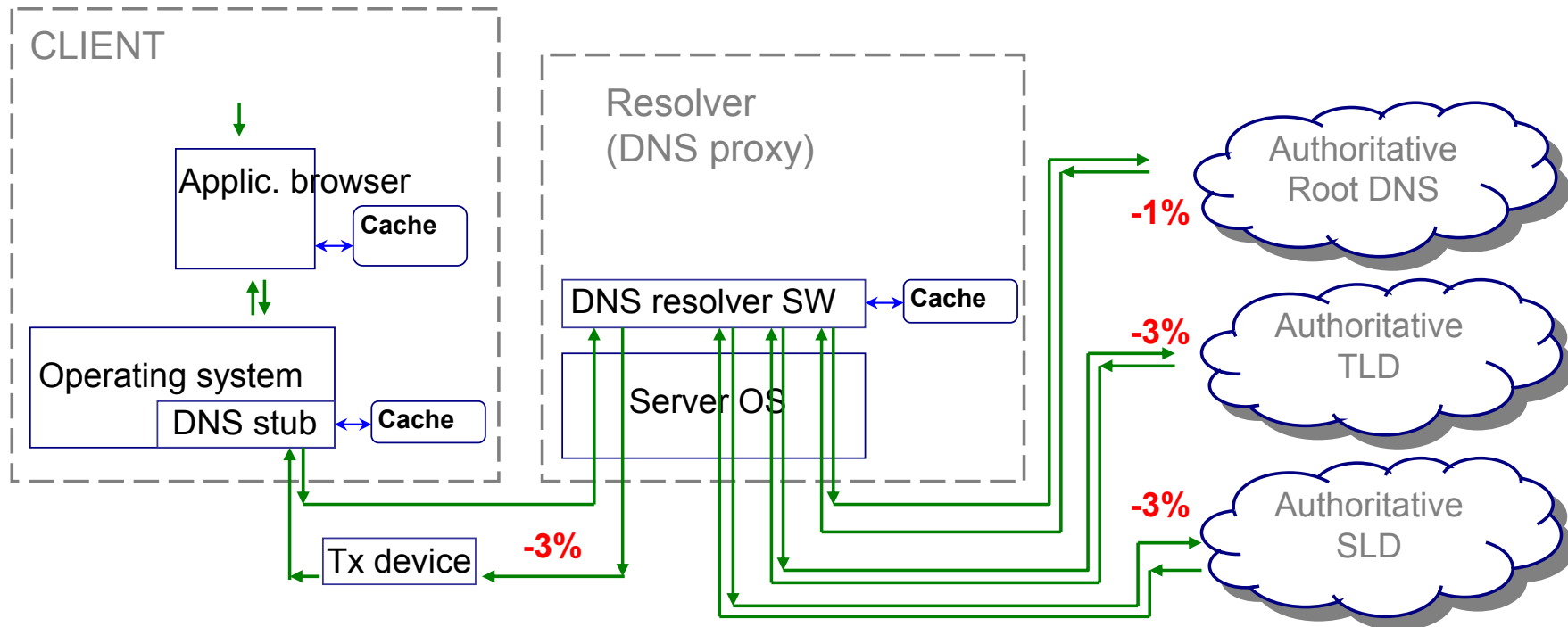




# Impact model (“perfect behavior”)



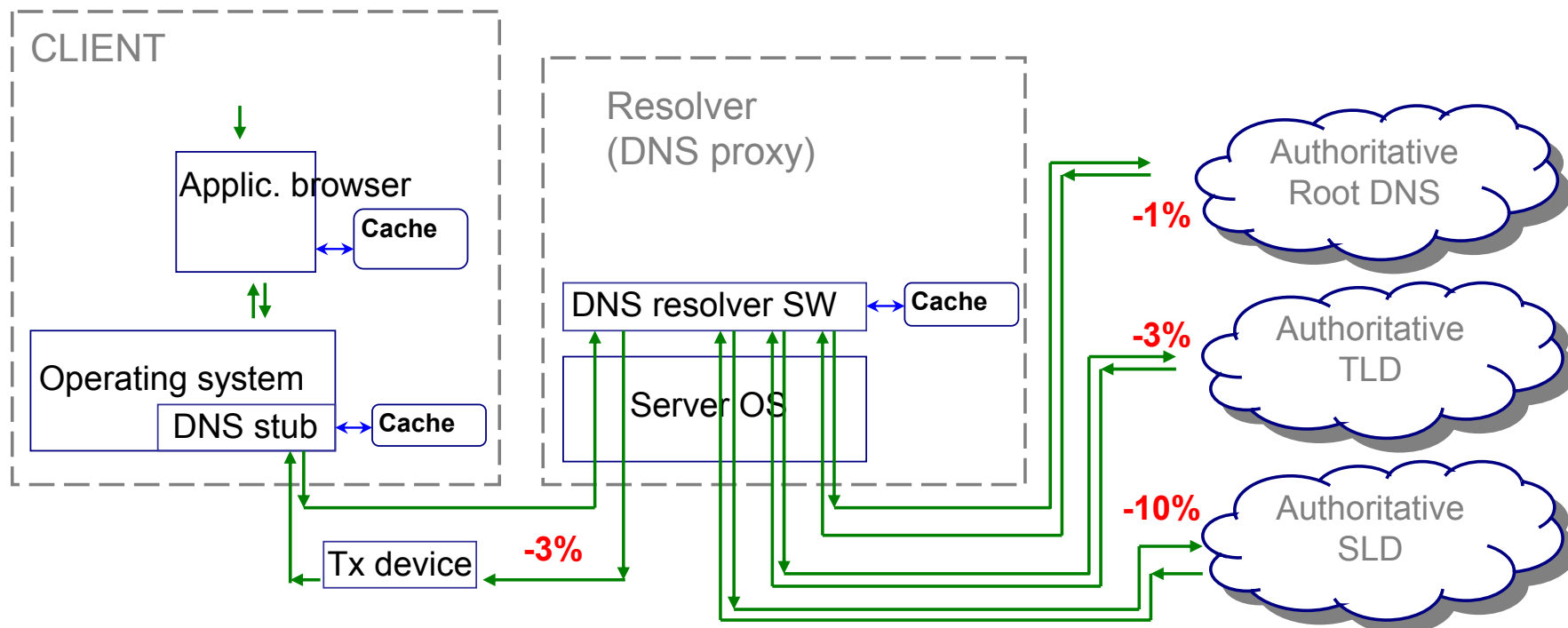
## Impact on average DNS traffic volume



- › **Predicted query load reduction as result of modifying aggressive Linux/Mac behavior is small**
  - › penetration of Linux / Mac OSX relatively low
  - › behavior occurs in case of 'exceptions' (ServFail, NXdomain, ...)

# Impact outlook

- scenario: 10% DNSSEC validation error for SLD

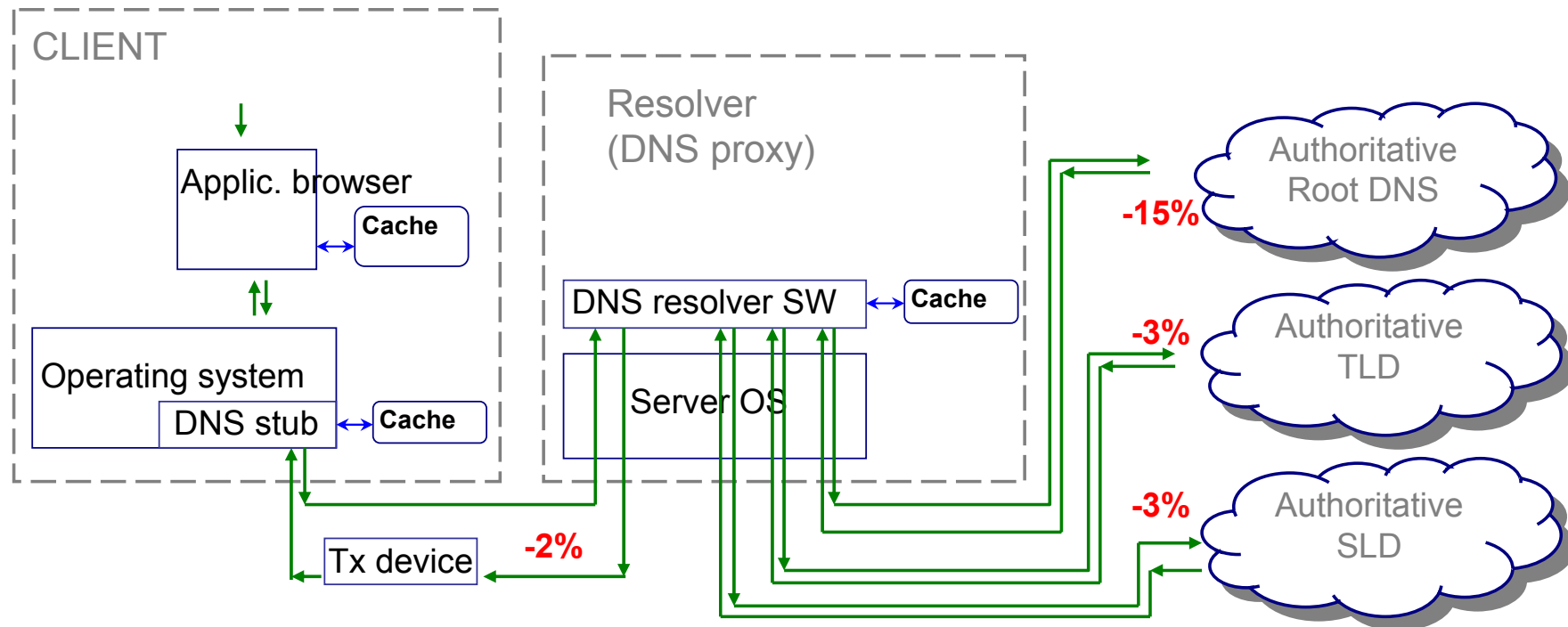


- › DNSSEC configuration errors at a domain will attract more traffic, due to observed behavior

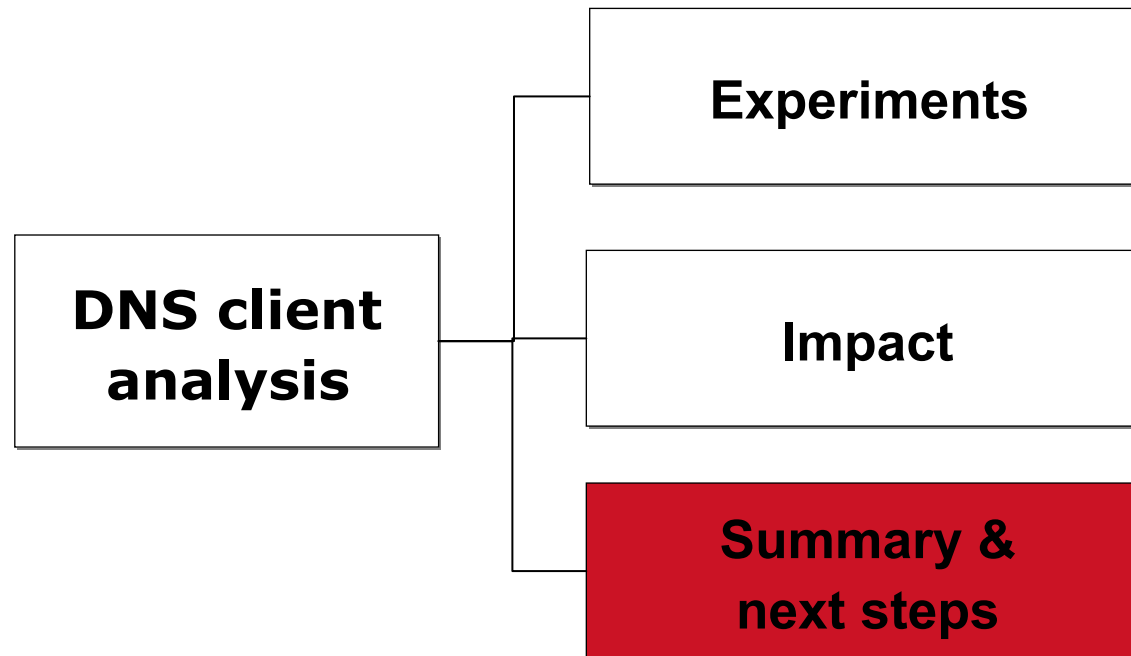


# Impact outlook

- scenario: NXdomain caching disabled at resolver



› Some amplification of bogus traffic to the Root



## Summary

- › Linux and Mac clients display aggressive DNS behavior, in case of non-valid responses
  - › Resolvers partly damp aggressive behavior, but also amplify it
- › Impact of client behavior on *average* DNS traffic is relatively low
  - › because fraction of Mac / Linux traffic is relatively low and
  - › behavior occurs in particular for minority of DNS responses
- › Although, for some particular cases the behavior amplifies traffic volume and rate

## Next steps

- › Share experiences with other experts
- › Contribute to improving DNS function in the glibc(?)
  - › alternative for pinpointed code part causing the amplification
- › Further quantitative scenario impact analysis
  - › further verification with ISP (SURFnet), SIDN data
  - › compare to greedy apps behavior
- › Is mobile internet different from other ISP traffic?
  - › ABI Research: “in 2015 62% of mobile device will be Linux-based” ...