DNS over TCP and TLS

draft-hzhwm-dprive-start-tls-for-dns-00

John Heidemann and Sara Dickinson

Joint work with Liang Zhu, Zi Hu,
Duane Wessels, Allison Mankin,
Willem Toorop

USC/ISI, Verisign Labs, and Sinodun
in collaboration with NLnet Labs, getdns

IETF91 / 11 November 2014

Our Goals

- DNS protocol changes
  - encouraging TCP
  - STARTTLS to initiate TLS
- implementation choices for good performance
- performance study to confirm costs
  - client latency: only modestly more
  - server memory: well within current hardware

Why DNS over TCP and TLS

- here: protecting privacy
  - encrypt stub-to-recursive queries

- use of TCP helps in other regards
  - defanging DoS
    - prevent attacks on the DNS server: use existing TCP anti-DoS (SYN cookies)
    - reducing attacks on others: TCP avoids amplification attacks
  - relaxing limits of UDP packet sizes: TCP

Protocol Changes: Goals

- minimize change
- reuse existing approaches
- follow IETF norms

(as boring as possible)

Protocol Changes: Goals

- minimize change
- reuse existing approaches
- follow IETF norms

(as boring as possible)

- **implications:**
  - **reuse TLS: Transport Layer Security**
  - **add a STARTTLS-like "upgrade"**
  - **dedicated port too, if that is acceptable under IANA Port Review (RFC 6335)**
  - **innovation: careful implementation**

SMTP before STARTTLS

C & S: open TCP connection

S: 220 mail.imc.org SMTP service ready

C: EHLO mail.example.com

S: 250-mail.imc.org hi, extensions are: -8BITMIME -STARTTLS DSN

problem: cleartext mail is snoop-able (fix: TLS)

C: MAIL FROM:<sender@mail.example.com>

S: 250 2.1.0 <sender@mail.example.com>... Sender OK

C: RCPT TO:<destination@mail.example.com>

S: 250 2.1.5 <destination@mail.example.com>

C: <send mail contents>

SMTP *with* STARTTLS (RFC-3207)

C & S: open TCP connection

S: 220 mail.imc.org SMTP service ready

C: EHLO mail.example.com

S: 250-mail.imc.org hi, extensions are: -8BITMIME -STARTTLS DSN

C: STARTTLS

S: 220 Go ahead

C & S: <negotiate a TLS session with a new session key, in binary>

C: EHLO mail.example.com

S: 250-mail.imc.org hello, extensions are: -8BITMIME DSN

C: MAIL FROM:<sender@mail.example.com>

S: 250 2.1.0 <sender@mail.example.com>... Sender OK

C: RCPT TO:<destination@mail.example.com>

S: 250 2.1.5 <destination@mail.example.com>

C: <send mail contents>

prologue: in clear (no privacy here)

transition to TLS

contents now private

this example: SMTP;
idea used for IMAP, POP3, FTP, XMPP, LDAP, NNTP…

Our STARTTLS for DNS

(draft-hzhwm-dprive-start-tls-for-dns-00)

C & S: open TCP connection

prologue

transition to TLS

C: QNAME="STARTTLS", QCLASS=CH, QTYPE=TXT
with the new TO bit set in EDNS options

S: RCODE=0, TXT="STARTTLS", with the TO bit set

C & S: <negotiate a TLS session, get new session key, in binary>

contents now private

C: <send actual query>

S: <reply to actual query>

pros: no new port (from IANA, or in firewalls)

cons: extra RTT; middleboxes may not like encrypted traffic

(other signaling approaches are possible)

Protocol Details

- keeps standard DNS framing before and after TLS upgrade
  - allows easy retrofit to existing resolver software
- use *dummy query* to avoid leaking information
- i-d says TO bit is *only* signaling
- pre-IANA, we use STARTTLS QNAME and no TO bit in our implementations

Our Goals

- DNS protocol changes
  - encouraging TCP
  - STARTTLS add TLS
- **implementation choices for good performance**
- performance study to confirm costs
  - client latency: only modestly more
  - server memory: well within current hardware

Careful Implementation Choices

- problem: no tuning of DNS TCP for queries *(until now!)*
  - see draft-dickinson-dnsop-5966-bis-00 (on DNSOP agenda today)
- connection reuse (or restart)
  - persistent connections
  - TCP fast open
  - TLS resumption
- query pipelining
- query reordering (out-of-order processing)

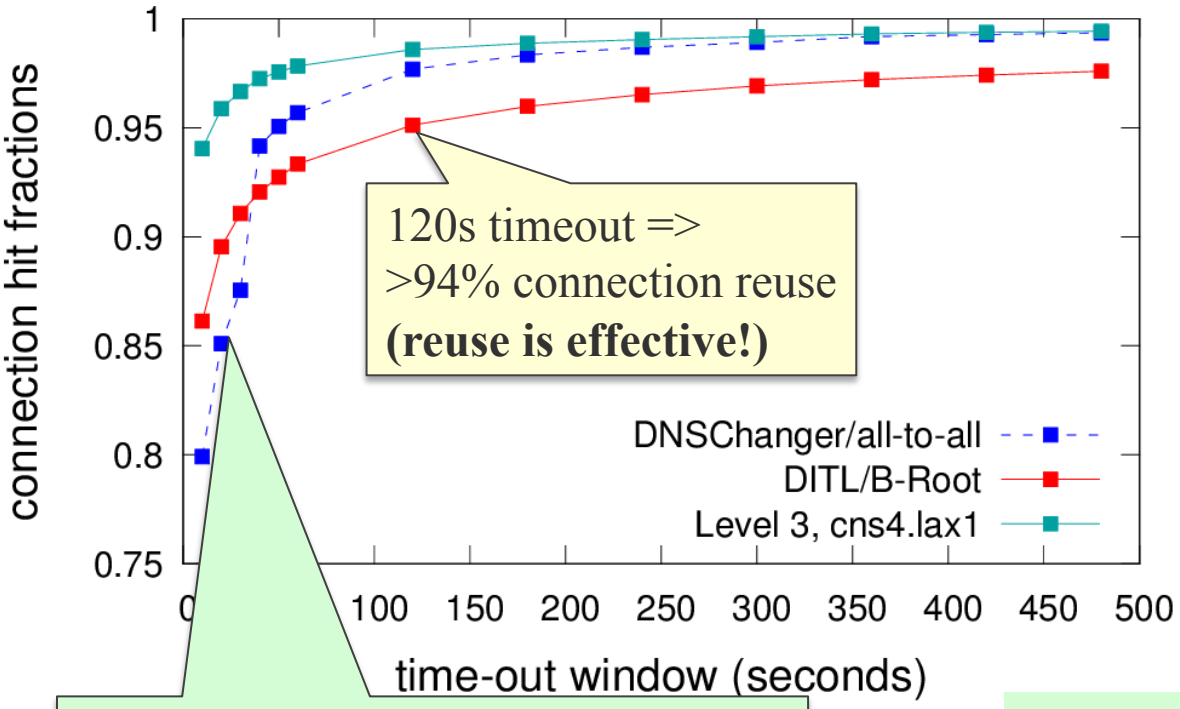
details in Sara's talk, and supplemental slides

Our Goals

- DNS protocol changes
  - encouraging TCP
  - STARTTLS add TLS
- implementation choices for good performance
- **performance study to confirm costs**
  - **client latency: only modestly more**
  - **server memory: well within current hardware**

details in tech report: "T-DNS: Connection-Oriented DNS to Improve Privacy and Security (extended)", ISI-TR-2014-693, http://www.isi.edu/~johnh/PAPERS/Zhu14b.pdf

Connection Reuse Helps? (YES!)



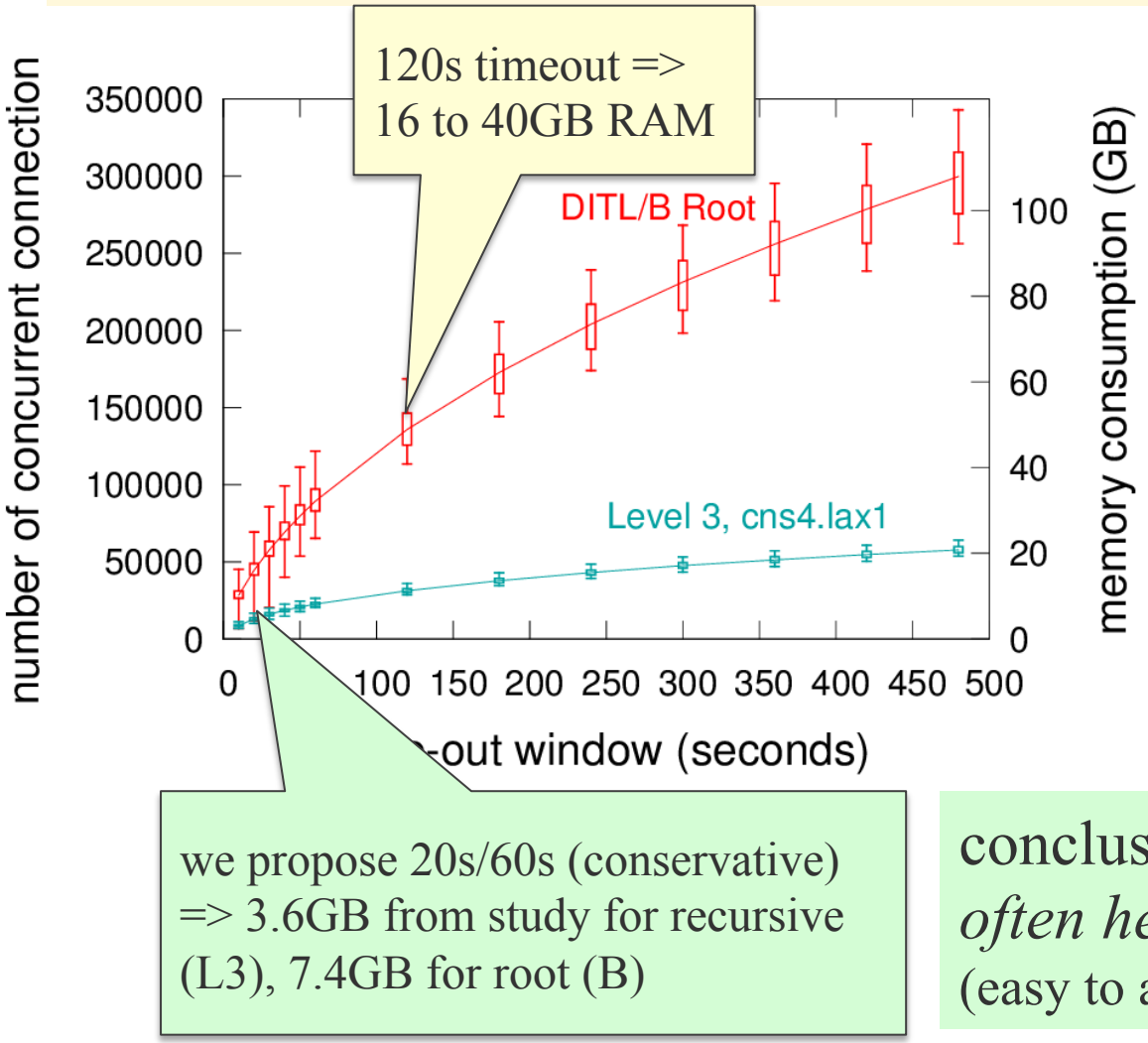
what fraction of queries find open TCP connections?

method: replay 3 traces: recursive (DNSchanger, Level3) and authoritative (B-Root)

(graph shows medians, quartiles are tiny)

conclusion: connection reuse is *often helpful*

Cost of Connection Reuse? (ok!)



we propose 20s/60s (conservative) => 3.6GB from study for recursive (L3), 7.4GB for root (B)

how many connections?
how much memory?

method: replay same 3 traces (here we show 2 biggest)

experimental estimate of memory: 360kB/connection (very conservative)

(graph shows medians and quartiles)

conclusion: connection reuse is *often helpful* and it's *not too costly* (easy to add server parallelism if needed)

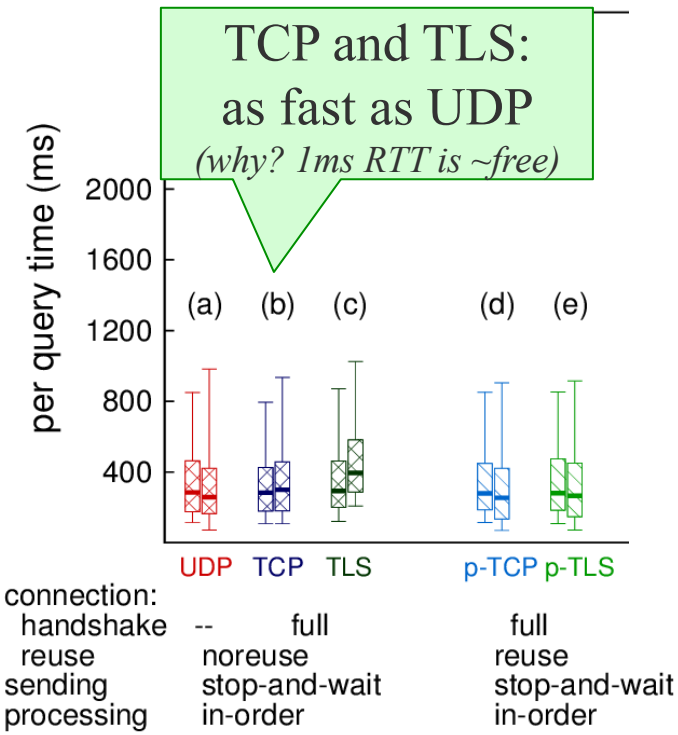
Latency: CPU Cost

- we used micro-benchmarks to study CPU cost

step	OpenSSL	GnuTLS
TCP handshake processing	0.15 ms	
TCP packet handling	0.12 ms	
TLS connection establishment	25.8 ms	8 ms
key exchange	13.0 ms	6.5 ms
CA validation	12.8 ms	1.5 ms
TLS connection resumption	1.2 ms	1.4 ms
DNS resolution (from 52)	0.1–0.5 ms	

TLS setup is noticeable,
but RTT (40-100+ms) more impt.

Latency: Stub to Recursive

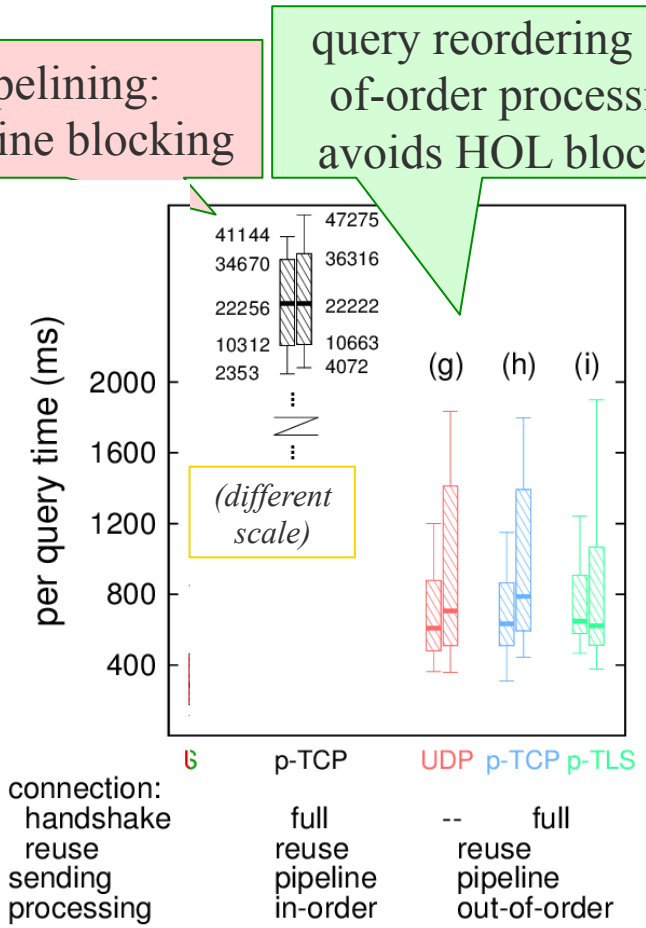


TCP and TLS vs. UDP? effects of implementation choices?
with short (1ms, left) and medium (35ms, right) RTTs

method: live experiments of random 140 names from Alexa top 1000; stub-recursive RTT=1ms

(graph shows medians and quartiles)

Latency: Stub to Recursive



TCP and TLS vs. UDP? effects of implementation choices?
with short (1ms, left) and medium (35ms, right) RTTs

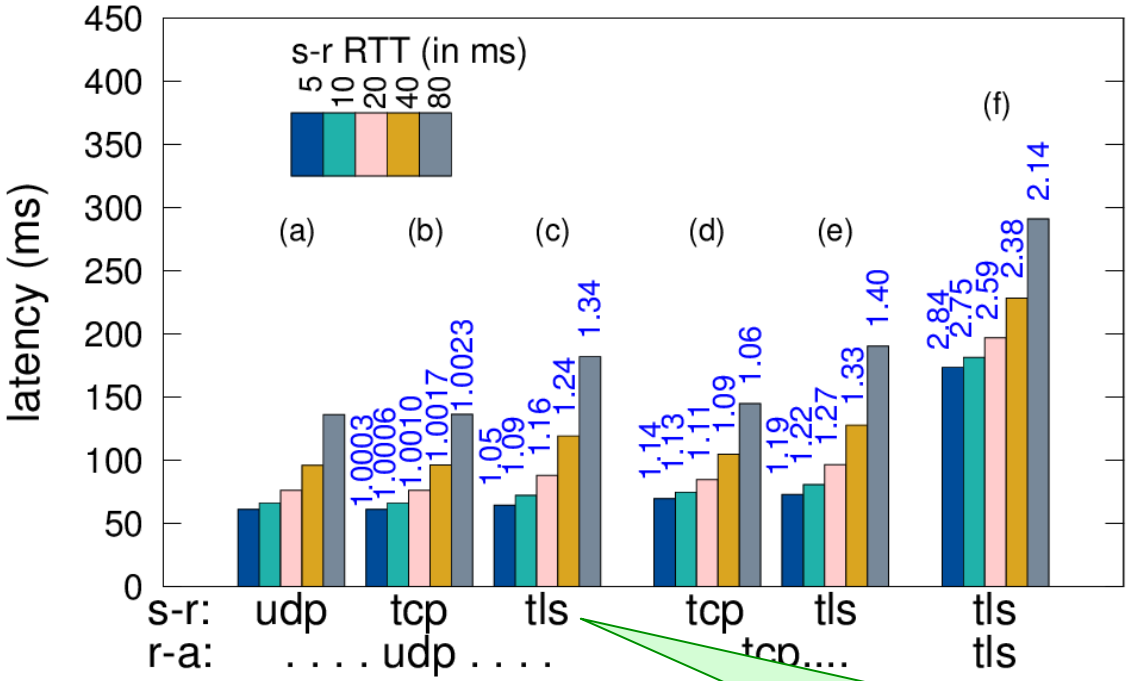
method: live experiments of random 140 names from Alexa top 1000

(graph shows medians and quartiles)

End-to-End Latency: Methodology

- controlled experiments are hard
  - variable stub query timing
  - caching at recursive resolver
  - different RTTs (many stubs and authoritatives)
- approach: *model expected latency*
  - i.e., just averages
  - median connection reuse from trace replay
  - other parameters from experiments

End-to-End Latency: Results



protocol choices: stub-recursive and recursive-authoritative

method: modeling; vary stub-recursive RTT; assumes all optimizations (TCP FO, TLS resumption, pipelining, OOOP)

(graph shows expected values, plus slowdown relative to case (a), UDP/UDP)

TLS (s-r, 60s t.o.) + UDP (r-a)
5 to 34% slower: **modest cost -> most benefit**

Our Goals

- DNS protocol changes
  - encouraging TCP
  - STARTTLS add TLS
- **implementation choices for good performance**
- performance study to confirm costs
  - client latency: only modestly more
  - server memory: well within current hardware

T-DNS Implementation Project Recap

- Aim: Running T-DNS code!
- People: Verisign Labs, Sinodun, NLnet Labs, getdns team, USC-ISI, …..
- Implementation Website:
  https://portal.sinodun.com/wiki/display/TDNS/T-DNS+Project+Homepage
- Past Presentations:

  DNSE at IETF89

  http://www.ietf.org/proceedings/89/slides/slides-89-dnse-3.pdf

  DNS-OARC Spring 2014 Workshop

  https://indico.dns-oarc.net//contributionDisplay.py?contribId=11&confId=19

Implementation Status

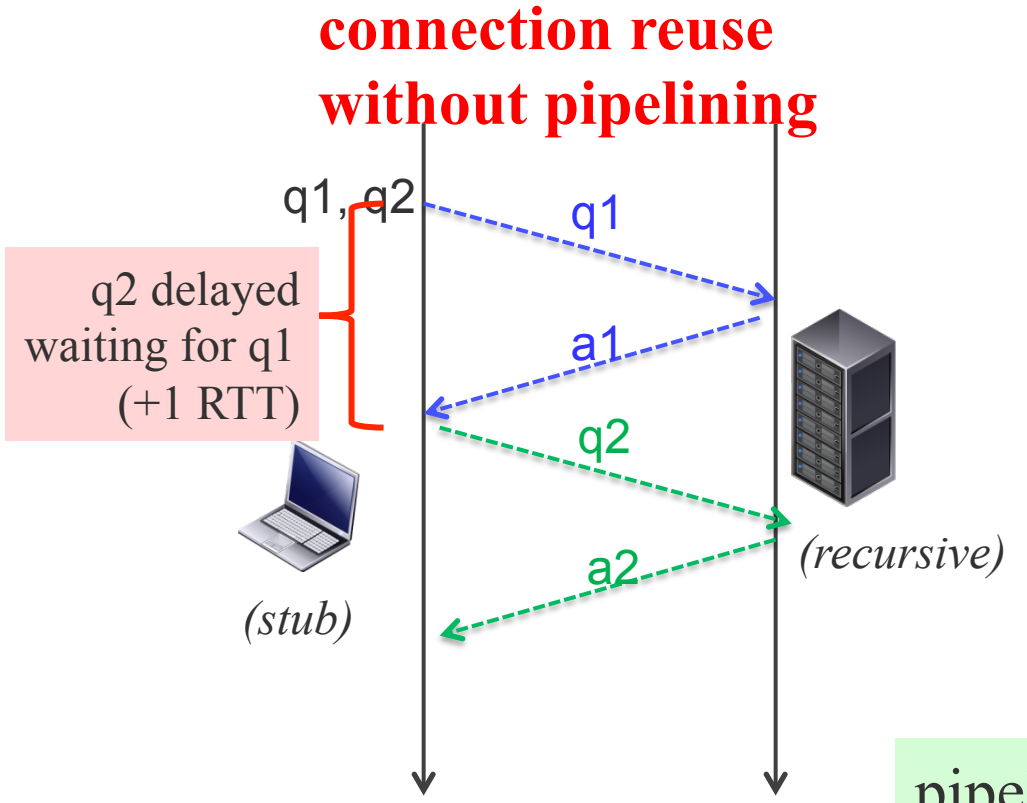
- initial prototyping
  - http://www.isi.edu/ant/software/index.html
  - digit: t-DNS client queries
  - (also client and server-side proxies; supports full protocol and cert authentication, but not for production use)
- current phase: targeting production software
  - LDNS (drill) / Unbound / NSD (NLnet Labs)
  - getdns (http://getdnsapi.net/)
- next phase includes BIND
- implementation notes
  - current code uses only dummy query (qname=STARTTLS, CH/TXT) to negotiate
    - use of TO bit pending IANA allocation
  - TLS-1.1 or better only (not SSL) as per UTA BCP
  - work-in-progress, still to do: certificate authentication

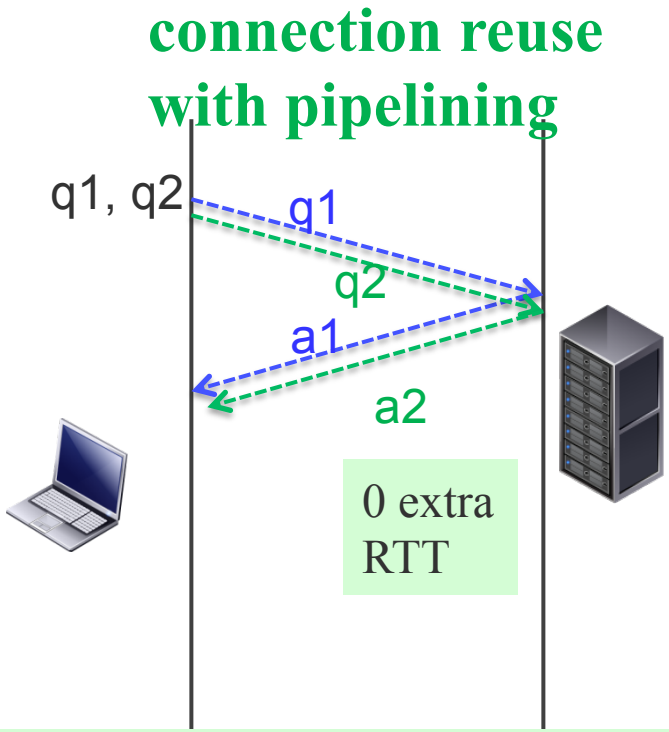
Performance and Functionality

- current focus: functionality
  - **T-DNS** (TLS)
  - TCP Fast open (reduces latency)
  - TCP connection re-use, and pipelining
  - query reordering (out-of-order processing)

Query Pipelining

send several queries immediately (not stop-and-wait)

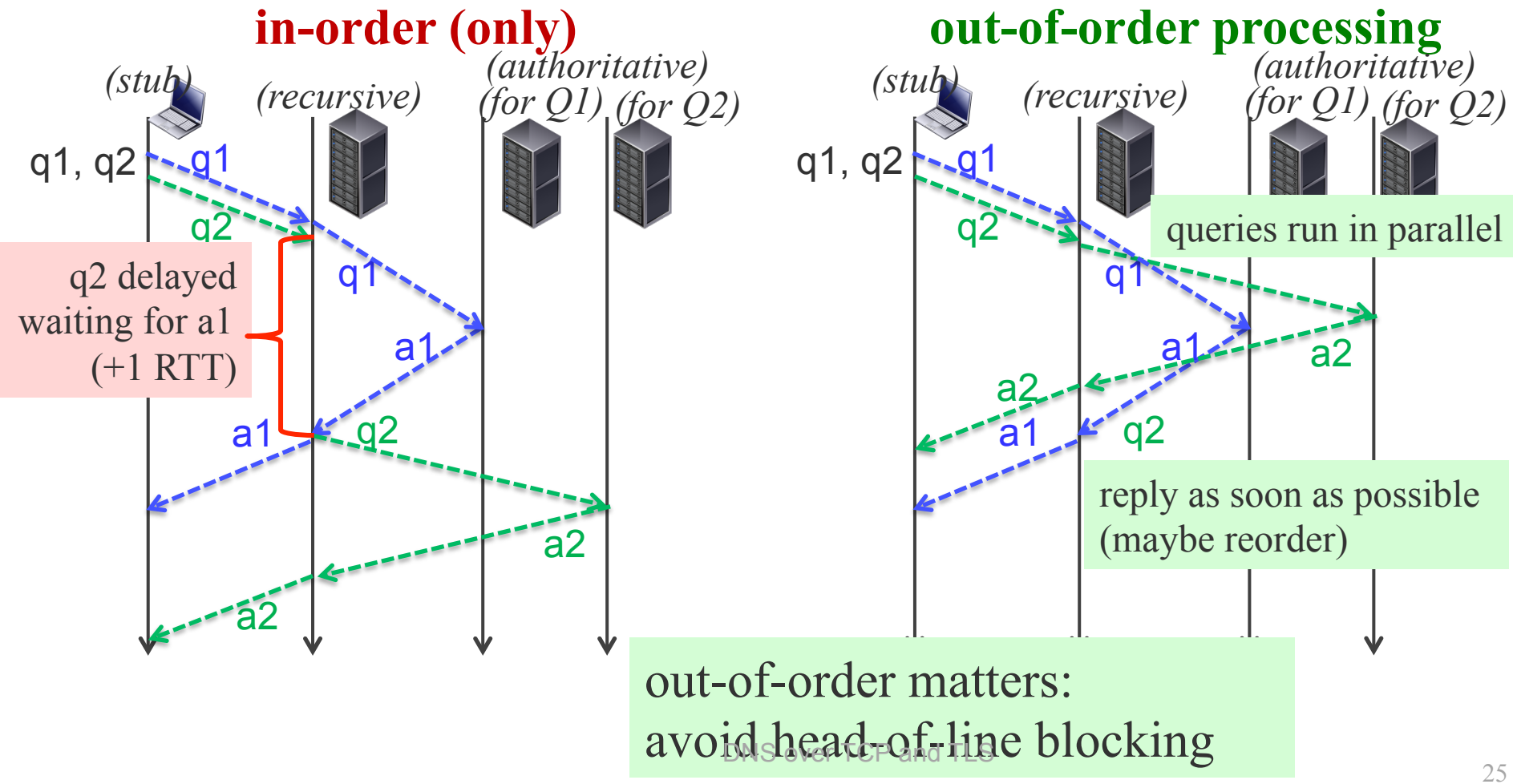




pipelining matters:
62% of web has 4+ domain names
(dataset: common crawl)

Out-of-Order Processing

process queries on same connection in parallel



Current Status (Detailed)

Software	digit	LDNS	getdns		Unbound		NSD
mode	client	client (drill)	stub	recursive*	server	client	server
TLS							
T-DNS							
TFO							
Conn reuse							
Pipelining							

Dark Green: Latest stable release supports this
Light Green: Patch available
Yellow: Patch in progress, or requires building a patched dependancy
Grey: Not applicable or not planned

** getdns in its recursive mode uses libunbound*

Demo Time

- patched version of *drill* querying patched *Unbound*
  - regular DNS UDP/TCP query
  - DNS query over TLS (dedicated port)
  - T-DNS (STARTTLS upgrade on TCP port 53)
  - [connection reuse/pipelining]
  - [TCP Fast Open]
    - STARTTLS goes in SYN; linux only
- wireshark screenshots at the end

T-DNS Next Steps

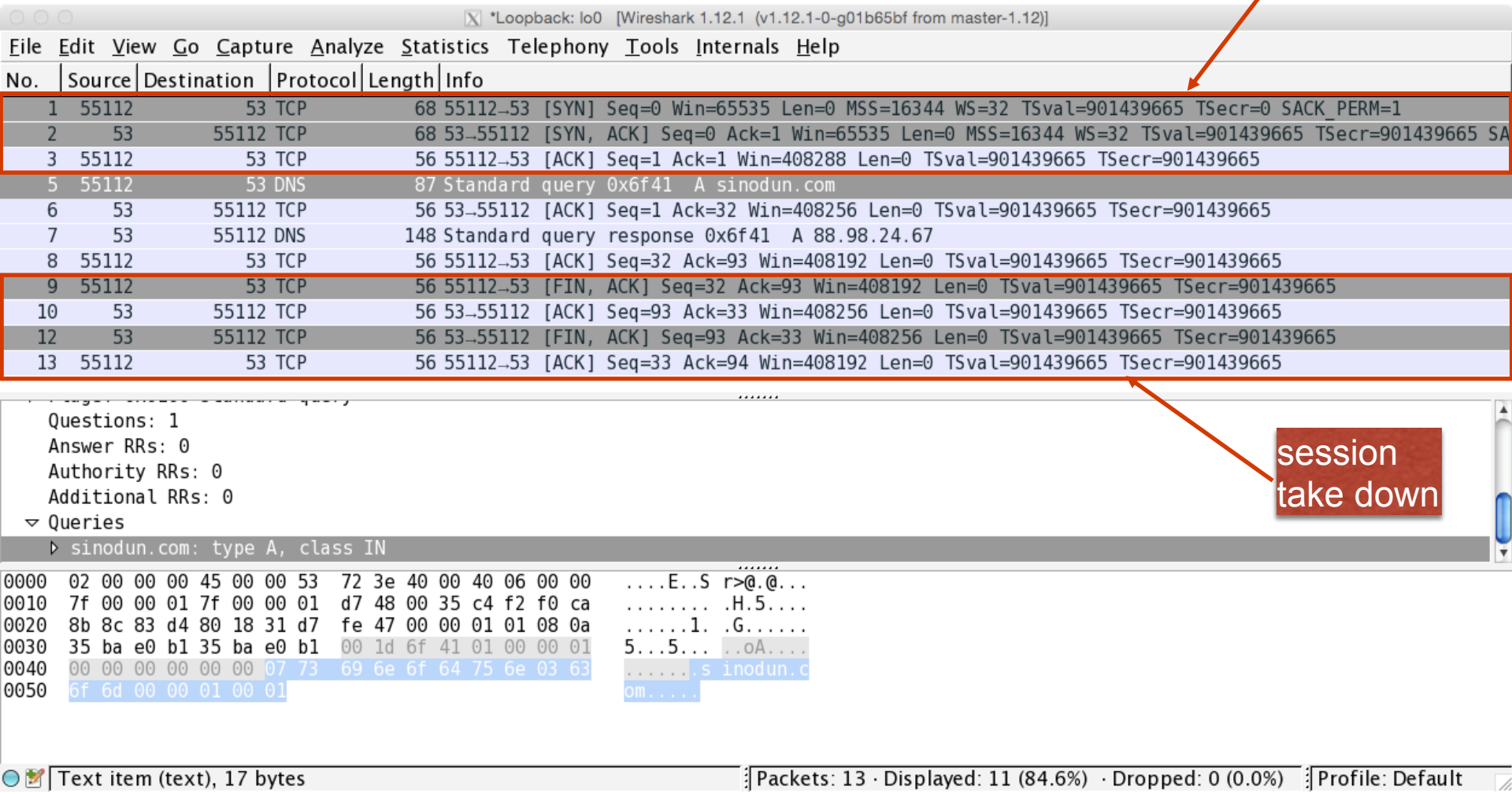
- more information:
  - tech report ISI-TR-2014-693 www.isi.edu/~johnh/PAPERS/Zhu14b/
- code:
  - client, client & server proxies, unbound patch
  - http://tdns.net
  - http://www.isi.edu/ant/software/
  - interoperability meeting tonight—come by for demo
  - working to get patches upstream
  - Bind implementation will begin next
- i-d for WG to consider adopting
  - draft-hzhwm-dprive-start-tls-for-dns-00

Appendices

Wireshark Screenshots

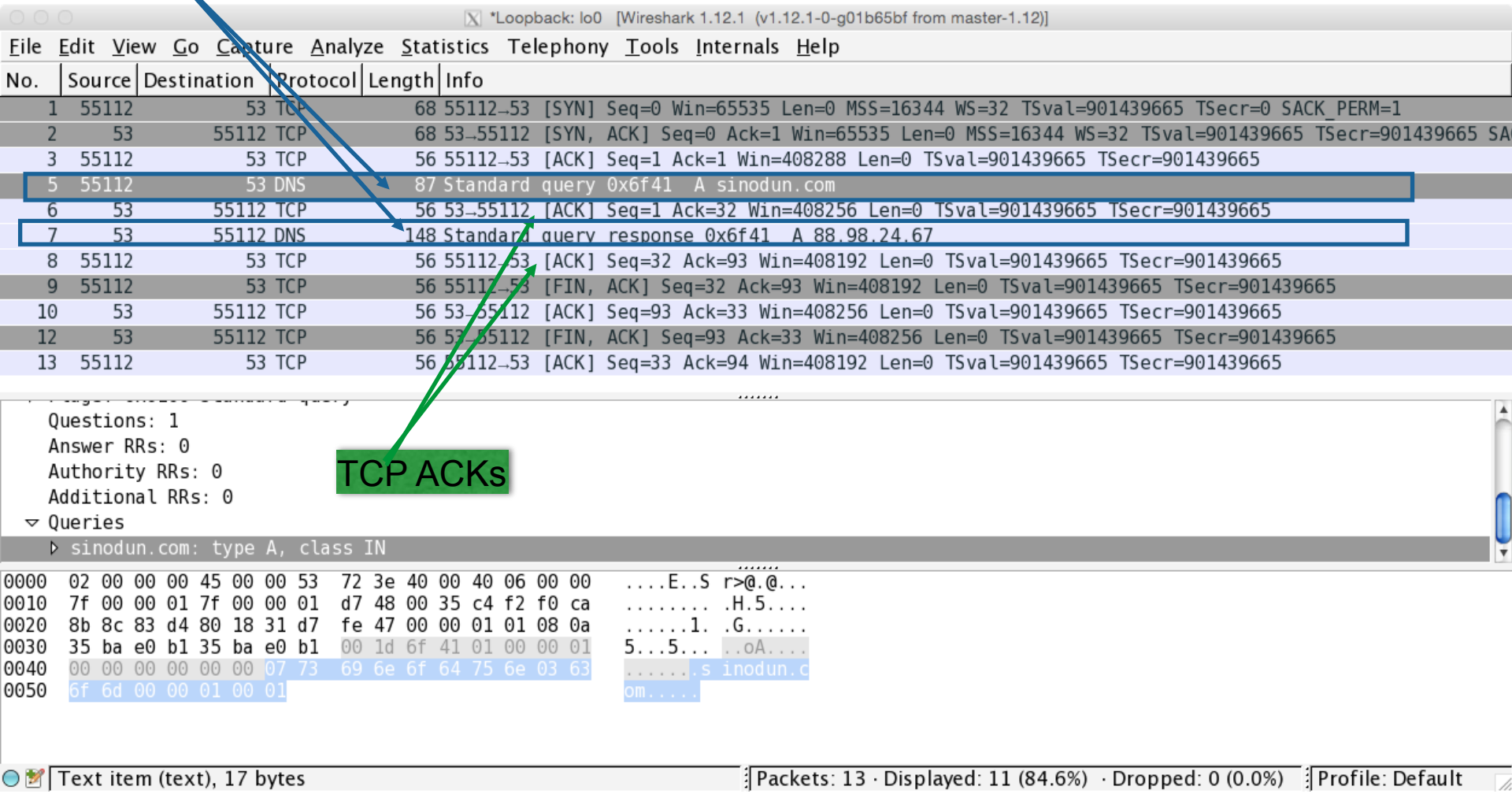
Supplemental Slides

TCP Query


3-way handshake
*Loopback: lo0 [Wireshark 1.12.1 (v1.12.1-0-g01b65bf from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
No. Source Destination Protocol Length Info
1 55112 53 TCP 68 55112→53 [SYN] Seq=0 Win=65535 Len=0 MSS=16344 WS=32 TSval=901439665 TSecr=0 SACK_PERM=1
2 53 55112 TCP 68 53→55112 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=16344 WS=32 TSval=901439665 TSecr=901439665 SA
3 55112 53 TCP 56 55112→53 [ACK] Seq=1 Ack=1 Win=408288 Len=0 TSval=901439665 TSecr=901439665
5 55112 53 DNS 87 Standard query 0x6f41 A sinodun.com
6 53 55112 TCP 56 53→55112 [ACK] Seq=1 Ack=32 Win=408256 Len=0 TSval=901439665 TSecr=901439665
7 53 55112 DNS 148 Standard query response 0x6f41 A 88.98.24.67
8 55112 53 TCP 56 55112→53 [ACK] Seq=32 Ack=93 Win=408192 Len=0 TSval=901439665 TSecr=901439665
9 55112 53 TCP 56 55112→53 [FIN, ACK] Seq=32 Ack=93 Win=408192 Len=0 TSval=901439665 TSecr=901439665
10 53 55112 TCP 56 53→55112 [ACK] Seq=93 Ack=33 Win=408256 Len=0 TSval=901439665 TSecr=901439665
12 53 55112 TCP 56 53→55112 [FIN, ACK] Seq=93 Ack=33 Win=408256 Len=0 TSval=901439665 TSecr=901439665
13 55112 53 TCP 56 55112→53 [ACK] Seq=33 Ack=94 Win=408192 Len=0 TSval=901439665 TSecr=901439665
session take down
Questions: 1
Answer RRs: 0
Authority RRs: 0
Additional RRs: 0
Queries
sinodun.com: type A, class IN
0000 02 00 00 00 45 00 00 53 72 3e 40 00 40 06 00 00 ....E..S r>@.@...
0010 7f 00 00 01 7f 00 00 01 d7 48 00 35 c4 f2 f0 ca .... .H.5....
0020 8b 8c 83 d4 80 18 31 d7 fe 47 00 00 01 01 08 0a ......1. .G......
0030 35 ba e0 b1 35 ba e0 b1 00 1d 6f 41 01 00 00 01 5...5... ..oA....
0040 00 00 00 00 00 00 07 73 69 6e 6f 64 75 6e 03 63 ....s inodun.c
0050 6f 6d 00 00 01 00 01 om.....
Text item (text), 17 bytes
Packets: 13 · Displayed: 11 (84.6%) · Dropped: 0 (0.0%)
Profile: Default

TCP Query

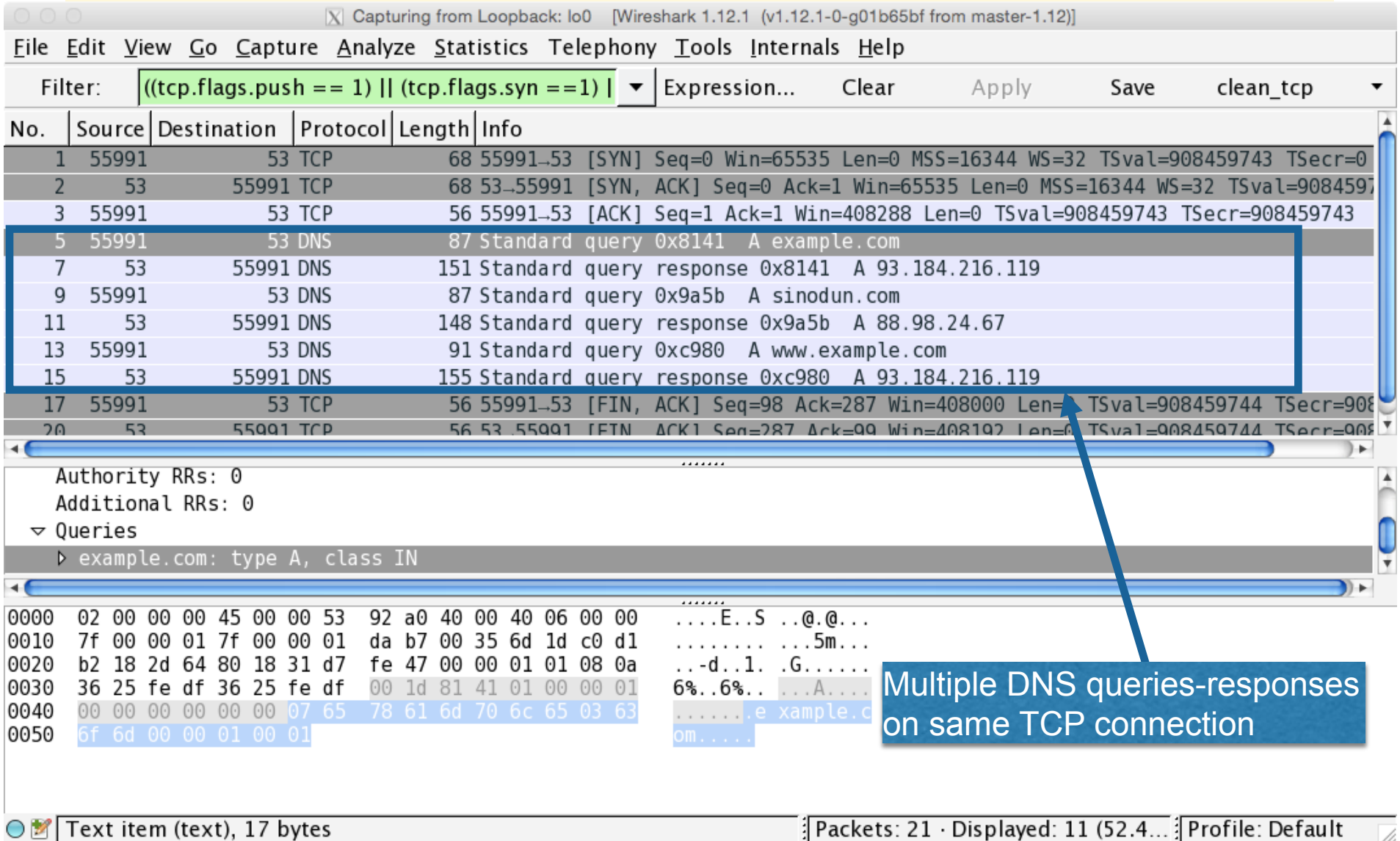
DNS query and response



TCP ACKs

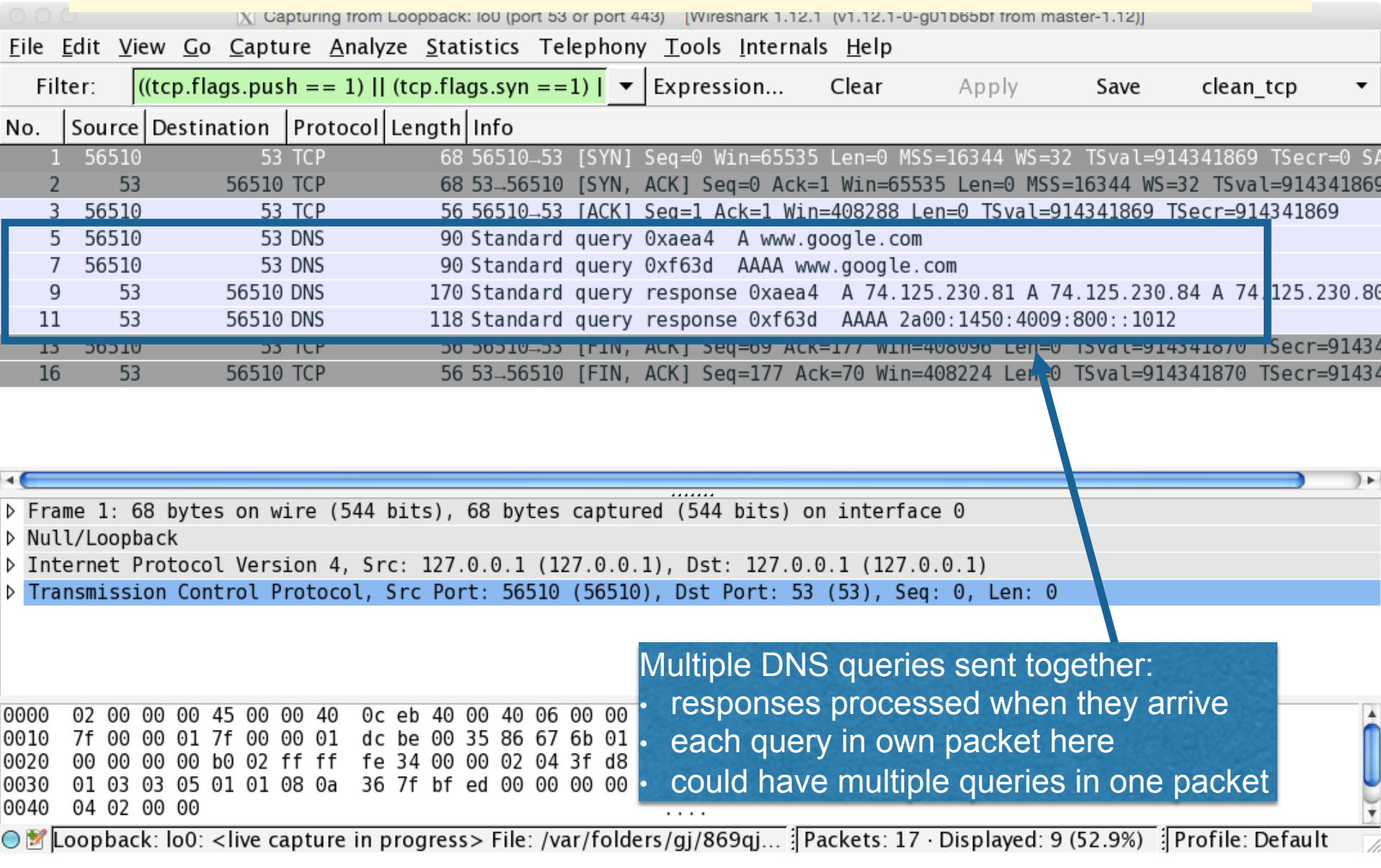
ACK filtered out

TCP connection re-use



ACK filtered out

TCP pipelining (getdns 0.1.5)



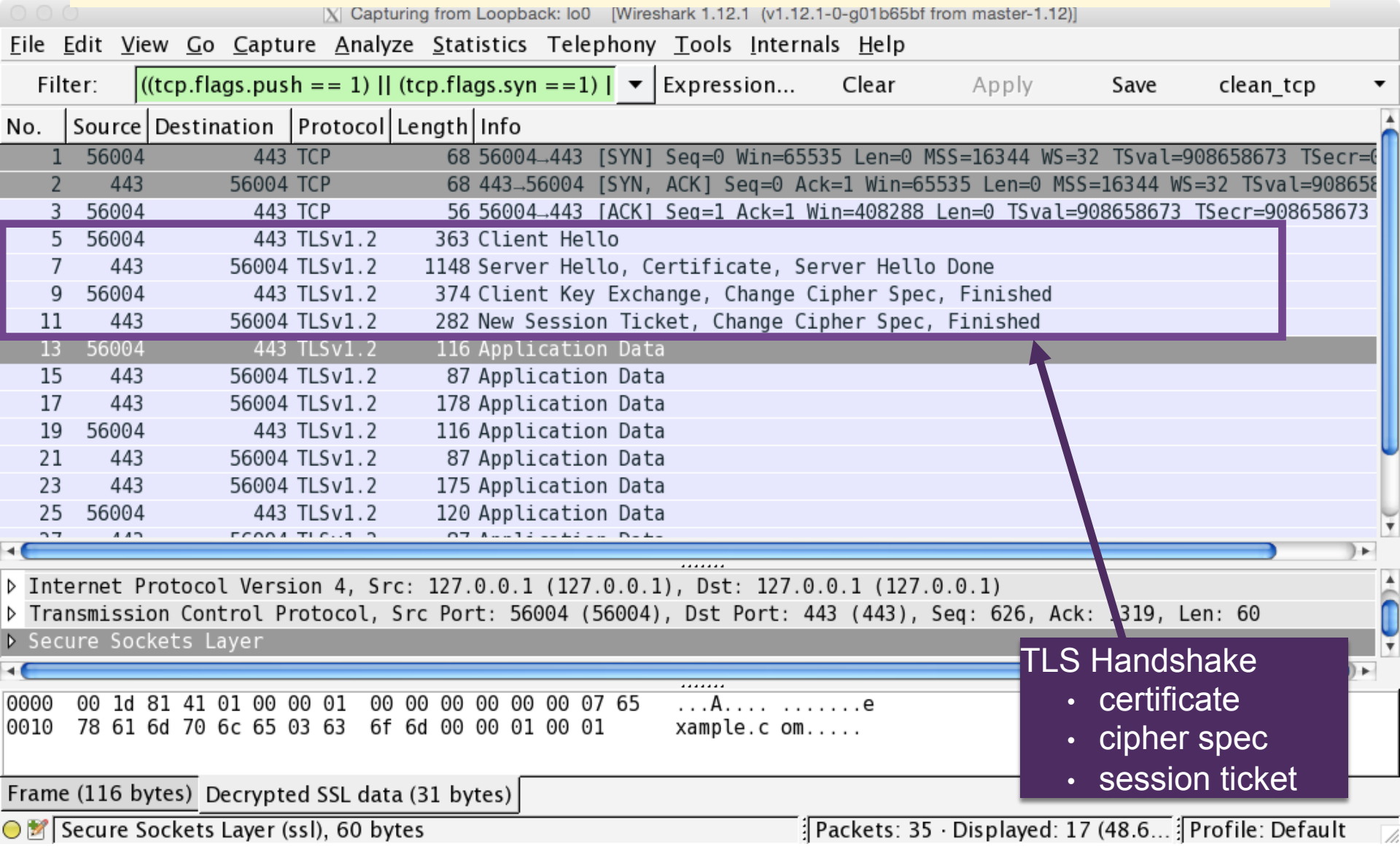
Multiple DNS queries sent together:

- responses processed when they arrive
- each query in own packet here
- could have multiple queries in one packet

Decode as SSL

ACK filtered out

TLS (port 443) - handshake



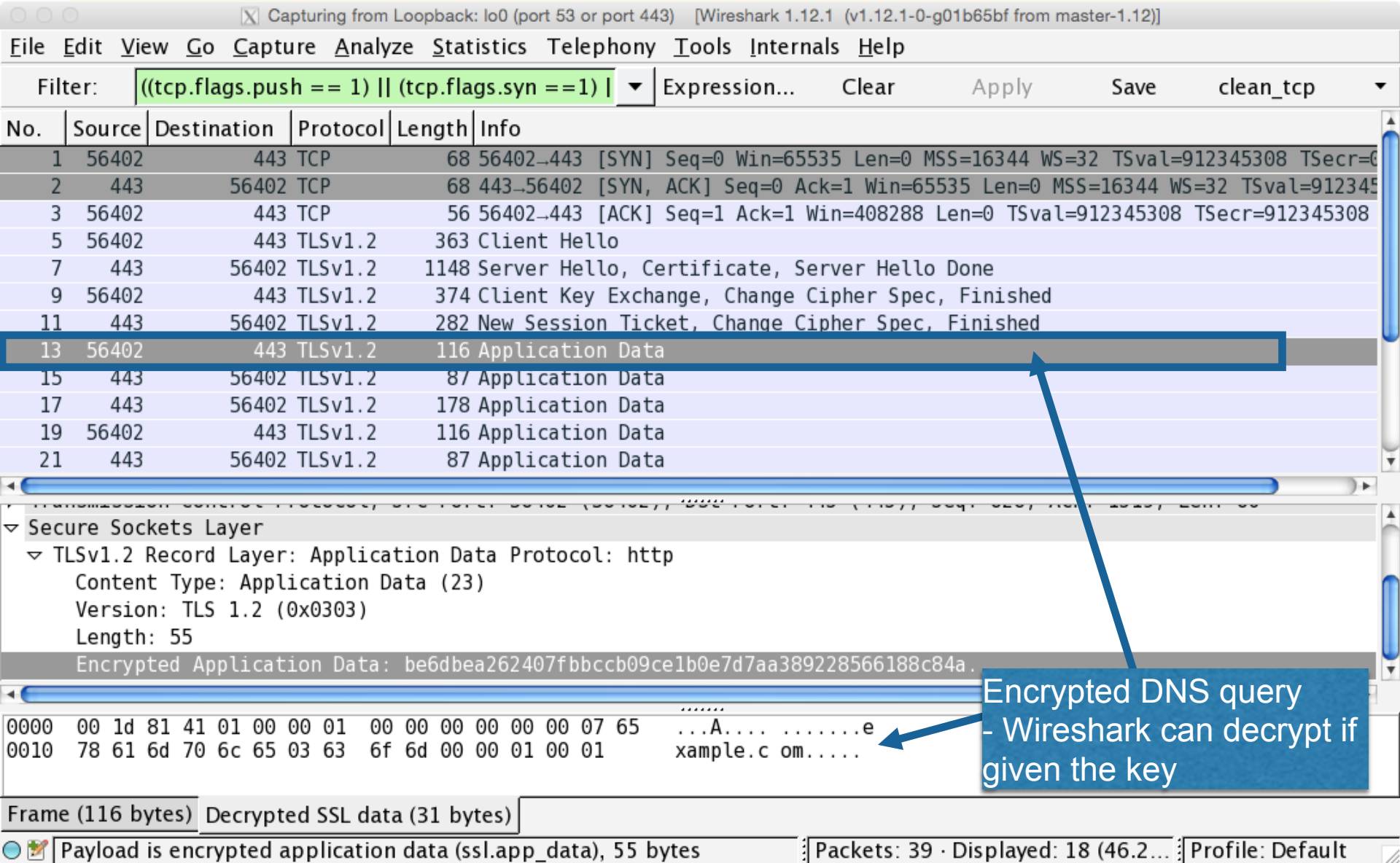
TLS Handshake

- certificate
- cipher spec
- session ticket

Decode as SSL

ACK filtered out

TLS (port 443) - DNS query

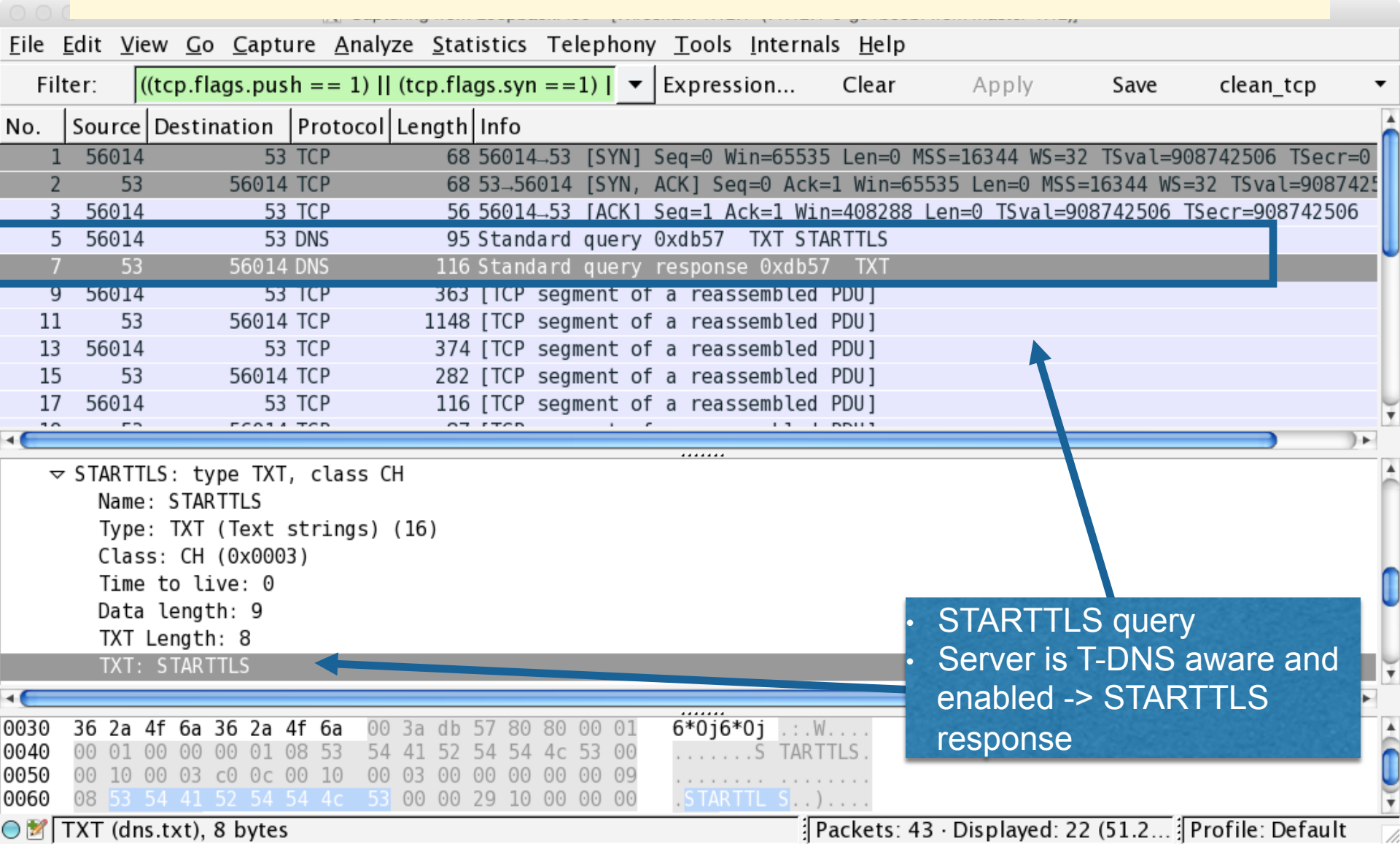


Encrypted DNS query - Wireshark can decrypt if given the key

Decode as DNS

ACK filtered out

T-DNS - STARTTLS dummy query

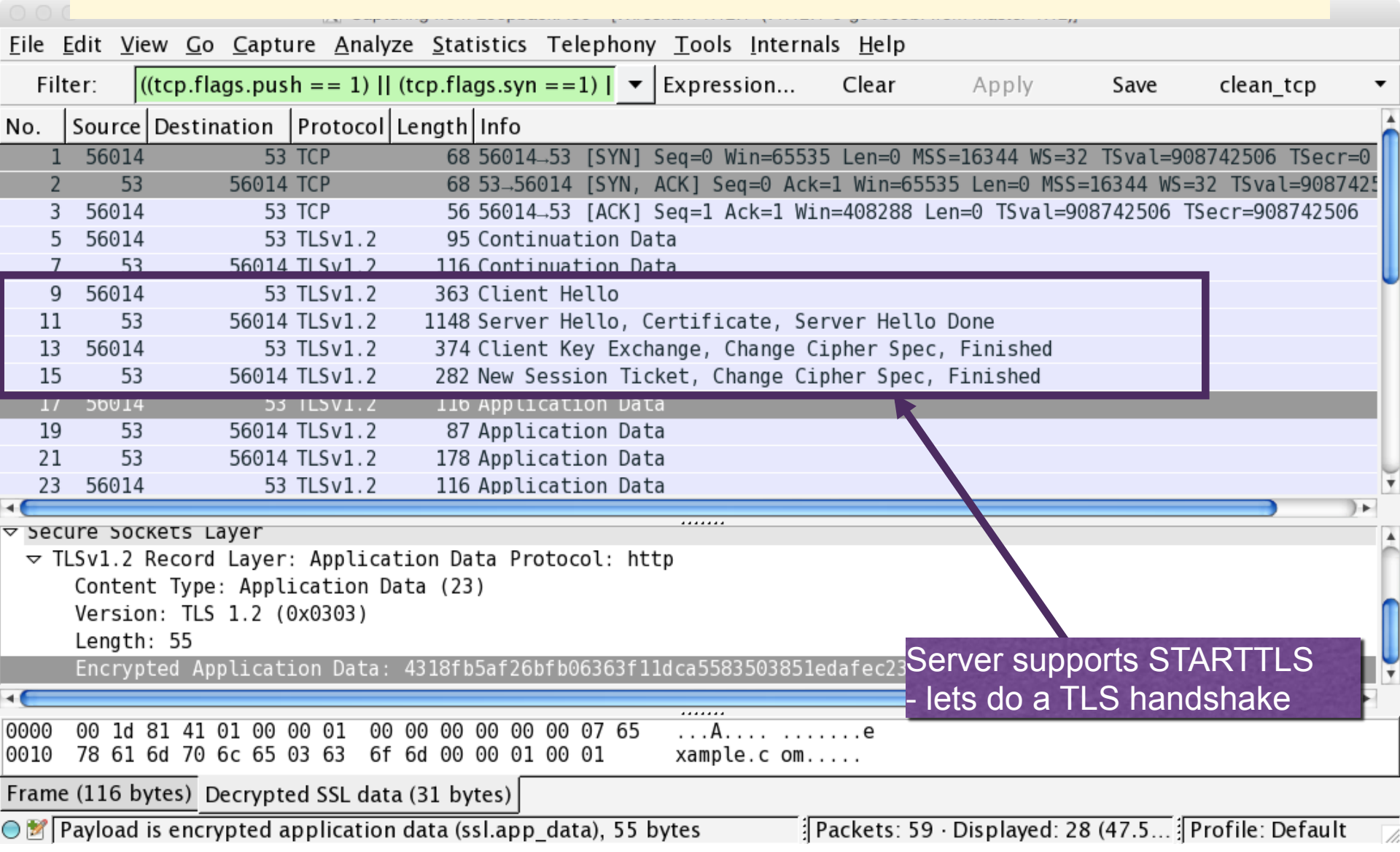


- STARTTLS query
- Server is T-DNS aware and enabled -> STARTTLS response

Decode as SSL

ACK filtered out

T-DNS - TLS handshake

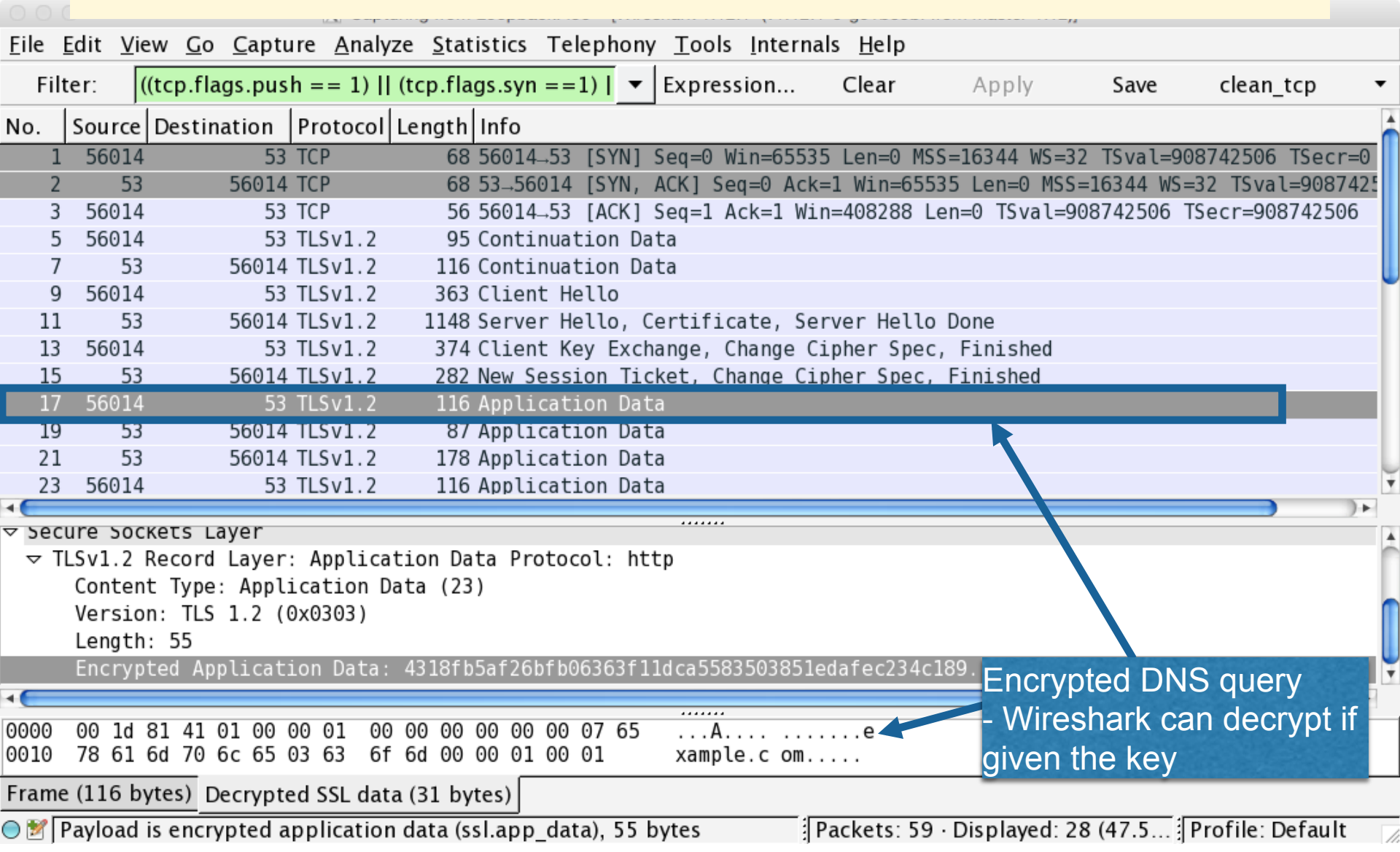


Server supports STARTTLS
- lets do a TLS handshake

Decode as SSL

ACK filtered out

T-DNS - DNS query

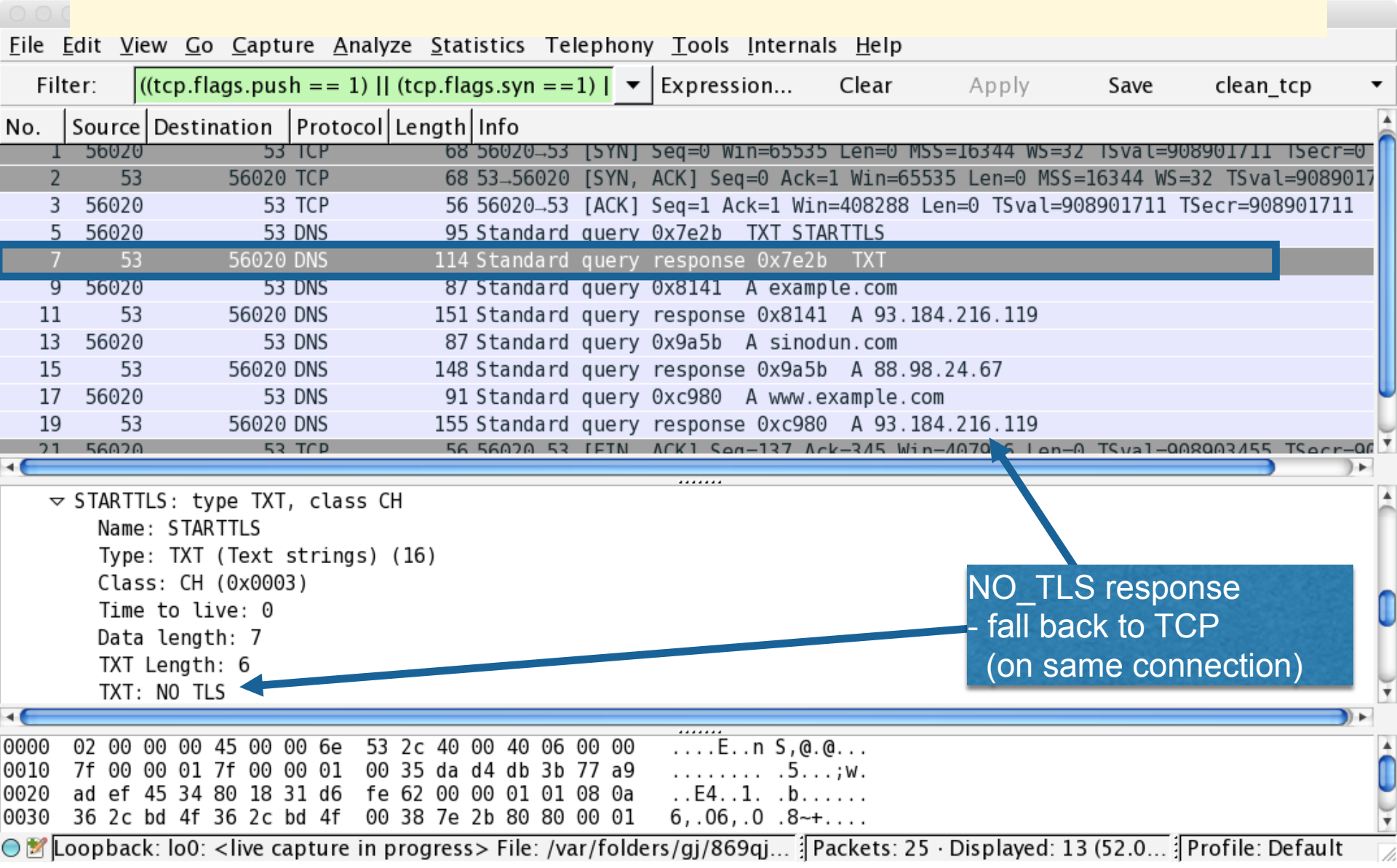


Encrypted DNS query
- Wireshark can decrypt if given the key

Decode as DNS

ACK filtered out

T-DNS - Fallback to TCP

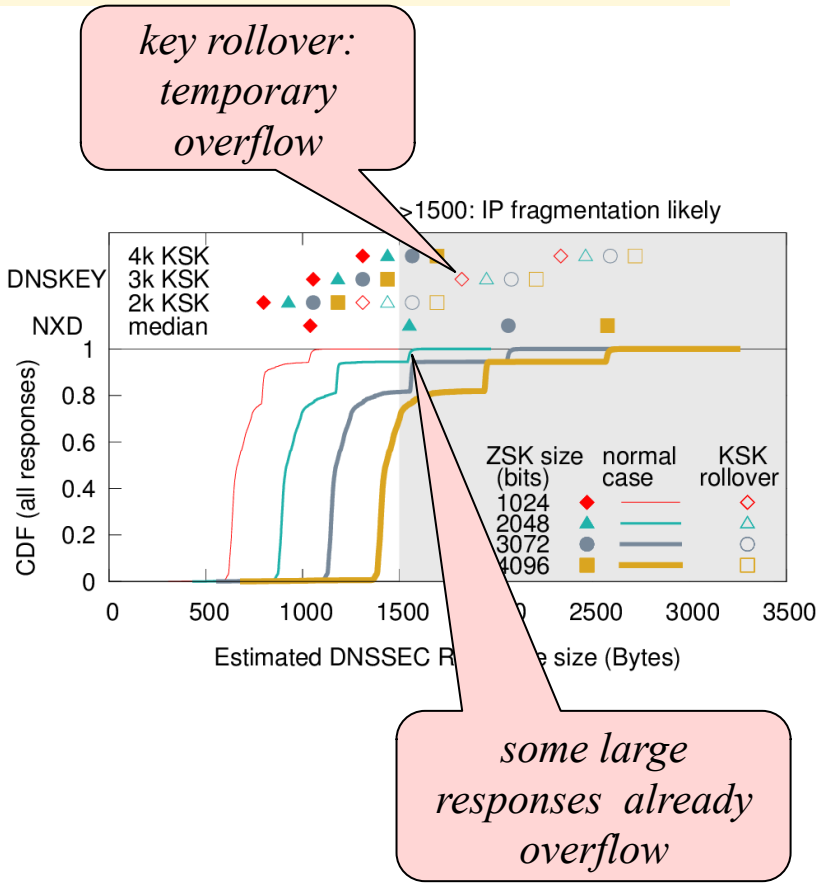


Supplemental Slides

- stresses on UDP-only DNS
- secure DNS relationship to TLS
- details about performance optimizations
- recursive-to-authoritative performance
- getdns

UDP Packet Size Limits

- for >25 years, *policy* decisions forced by UDP packet sizes
  - number of root servers: all fit in 512B
  - DNSsec: required EDNS
  - crypto algorithm and key size
- partial fix: EDNS0 deployment (10+ years, since 1999)
- but packet size lurks
  - keysizes
  - IPv6 records
  - certs in DNS (for DANE)



Doesn't DNSsec already *"Secure DNS"?*

A: yes, but…

- DNSsec is about *query integrity*
  - that is: if you are told X, is X true?
  - it signs answers; signatures prove X is true
- DNSsec does *nothing* for privacy and DoS
  - *everything* sent in the clear: *no privacy*
  - nothing about DoS
  - large signatures stress UDP size limits

=> need DNSsec's integrity ***and*** T-DNS' privacy

Latency in DNS/TLS

C & S: open TCP connection

TCP 3wh: +1 RTT

STARTTLS: +1 RTT

C: QNAME="STARTTLS", QCLASS=CH, QTYPE=TXT
with the new TO bit set in EDNS options

S: RCODE=0, TXT="STARTTLS" with the TO bit set

C & S: <negotiate a TLS session with a new session key. in binary>

TLS handshake: +2 RTTs

C: <send actual query>

S: <reply to actual query>

query: 1 RTT

Connection Reuse

- basic idea:
  reuse connection -> no setup cost

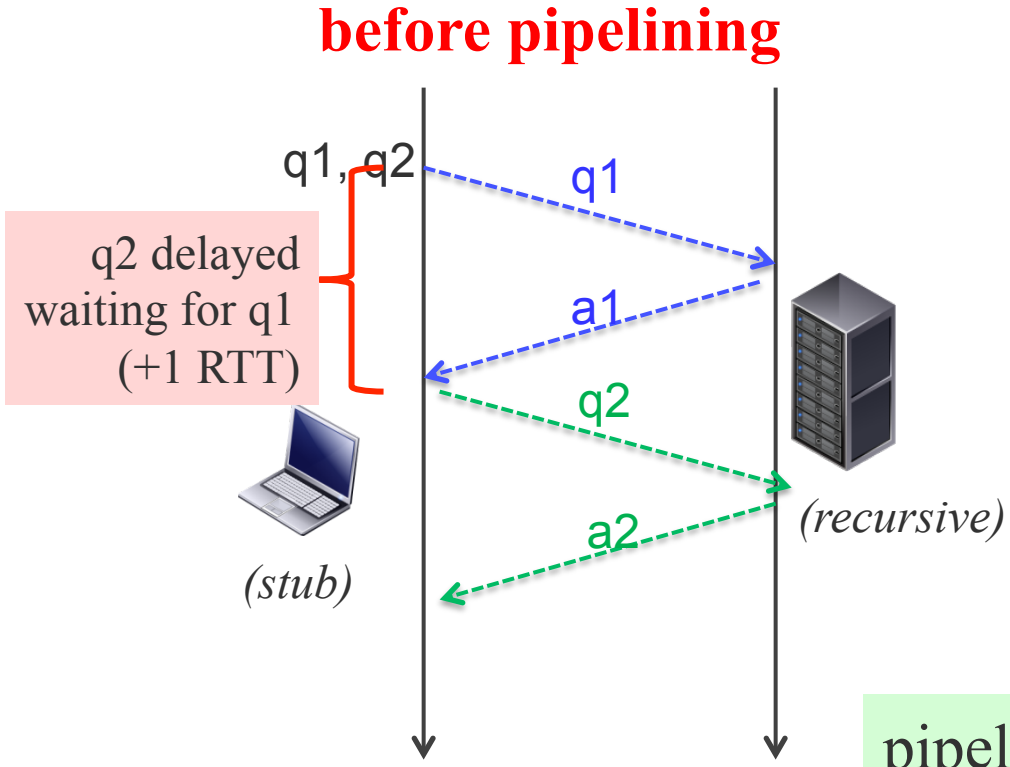
- secondary idea:
  if must close, client keeps state to restart quickly

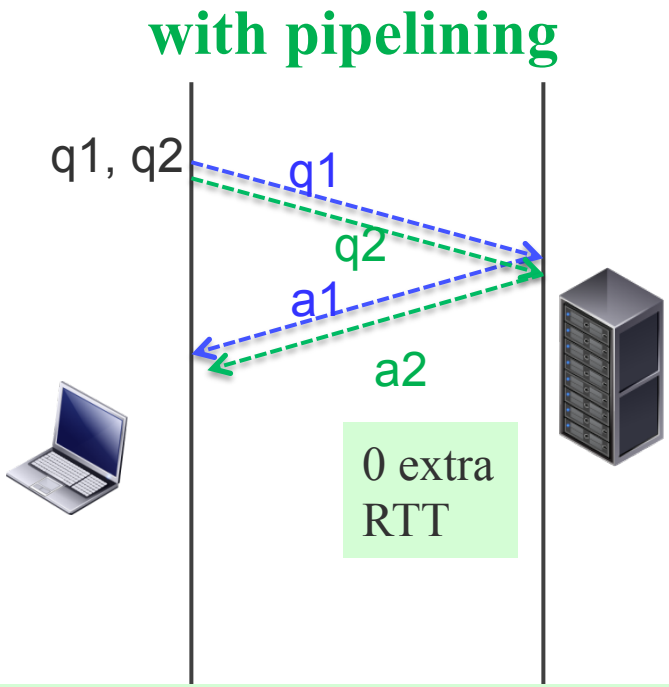
Connection Reuse

- basic idea:
  reuse connection -> no setup cost
  - *persistent connections (in client and server)*
- secondary idea:
  if must close, client keeps state to restart quickly
  - *TCP fast open: client has cookie to send data in 3wh*
    - *draft-ietf-tcpm-fastopen-08: in Linux-3.6, default 3.13*
  - *TLS resumption (RFC-5077): client keeps*
    - *RFC-5077: in OpenSSL and GnuTLS*

Query Pipelining

send several queries immediately (not stop-and-wait)





pipelining matters:
62% of web has 4+ domain names
(dataset: comanon crawl)

(Digression) DNS Resolution: stub -> recursive -> authoritative

stub

at end-user

generates queries, recursive does work

Q: A www.example.com? -> rec

recursive

at user's ISP *or* public DNS in a CDN

converts user query to many authoritatives; caches replies

Q: A www.example.com? -> .
Q: A www.example.com? -> .com
Q: A www.example.com?
-> example.com
A: 192.0.52.1

authoritative

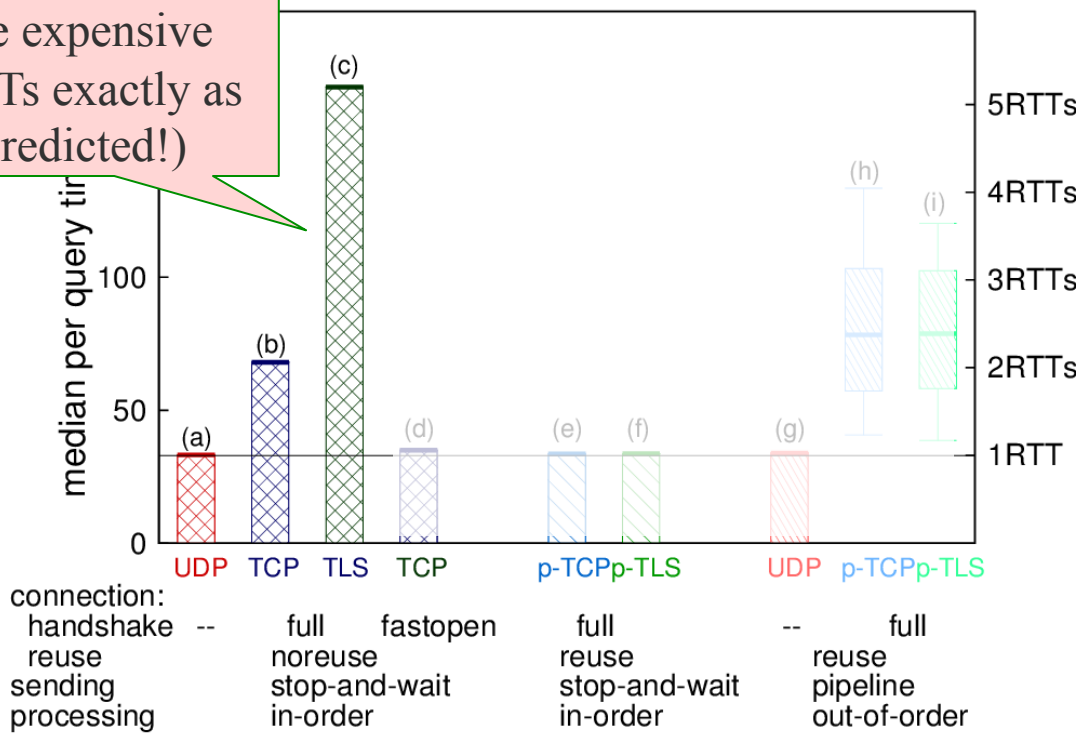
at provider (maybe replicated)

has actual answers

A: see NS for .com
A: see NS for example.com
A: 192.0.52.1

Latency: Recursive to Authoritative

new connections are expensive (RTTs exactly as predicted!)

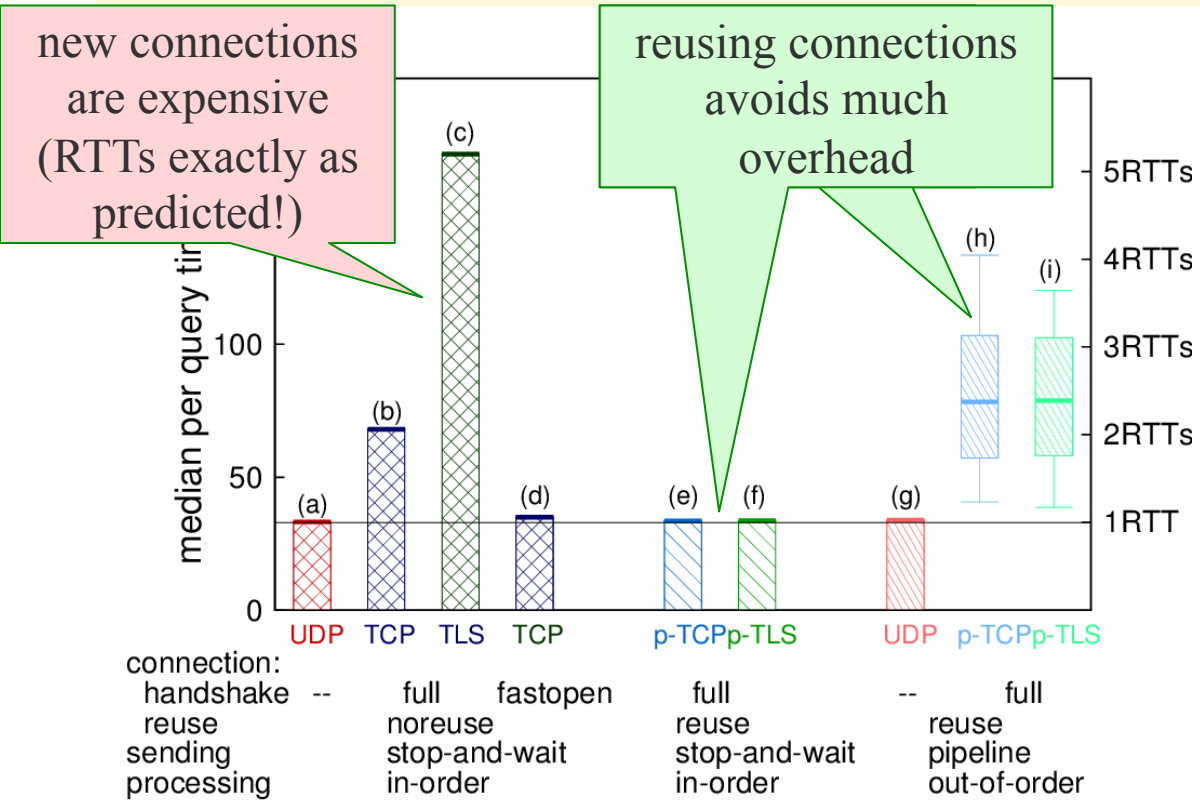


TCP and TLS vs. UDP? effects of implementation choices?
*with **long** RTT (=35ms)*

method: live experiments of random 140 names, each repeaed 10x; recursive-authoritative RTT=35ms

(graph shows medians and quartiles for (h) and (i), or bars where median and quartiles are the same)

Latency: Recursive to Authoritative



TCP and TLS vs. UDP? effects of implementation choices?
*with **long** RTT (=35ms)*

method: live experiments of random 140 names, each repeaed 10x; recursive-authoritative RTT=35ms

(graph shows medians and quartiles for (h) and (i), or bars where median and quartiles are the same)

getdns

- getdns API (http://getdnsapi.net/)
  - modern, asynchronous DNS API specification
  - API originally by Paul Hoffman
  - created by and for application developers
- getdns is the first (and currently only) implementation of this specification
- open source C implmentation developed and maintained in collaboration by NLnet Labs, Verisign Labs, and No Mountain Software