# T-DNS: Connection Oriented DNS to Improve Privacy and Security

John Heidemann[1]

joint work with Liang Zhu[1], Zi Hu[1], Duane Wessels[2], Allison Mankin[2], Nikita Somaiya[1]

[1:] USC/ISI, [2:] Verisign Labs

10 May 2014



---

# don't fear connections for DNS

---

## DNS Basics

since 1987 (RFC-1034)
DNS is simple request-response:

client: A www.example.com ?

*server: 192.0.2.1*

perfect for UDP
(TCP supported too, but as fallback and zone transfers)



---

## Fear of DNS over TCP

- TCP is horribly slow: *bad client latency*
- TCP => server state : *server memory explodes*

community ~~consensus:~~ ~~orthodoxy~~ ***dogma***
*don't use TCP**, UDP's constraints are OK

* except for fallback and zone transfers



---

## Our Contributions

- analysis: **don't fear connections for DNS**
  - client latency: only modestly more
  - server memory: well within current hardware
- implementation choices to get here
- small protocol addition: TLS upgrade

=> *T-DNS: DNS over TCP+TLS*



---

## T-DNS: TCP and TLS Connections

- introduction
- **why**
- how
- at minimal cost
- better than alternatives
- next steps

## Why T-DNS

- protecting privacy
  - connections -> encryption -> privacy
- denying DoS (Denial of Service)
  - connections -> spoof-proof -> no amplification attacks
- leaving limits
  - connections -> UDP limits don't drive policies



## Protecting Privacy

- principle: *all* traffic should be private (=> encrypted)
- rise of public DNS means many can snoop
  - Google Public DNS, OpenDNS, others
  - traffic over WAN should be private!
- individuals avoiding transparent proxies
  - multiple ISPs intercept DNS to add ads
- DNS is more than addresses
  - anti-spam (DNSBL), embedded user IDs (facebook, etc.)
    - ex: DNSBL's spam check sends IP address of *every incoming mail server* over the WAN
  - even on LAN (where destinations are visible), should protect other content


*advocacy of Google public DNS to avoid Turkish censorship of Twitter, 2014-03-21*



## Denying DoS

- problem: DNS attacks others
  - DNS amplification attacks
  - a growth industry in 2013: >100Gb/s attacks
- problem: DoS on DNS servers
  - work-around: massive over-capacity
- solution: TCP
  - well understood anti-DoS methods:
  - 3-way handshake precludes spoofing
  - TCP cookies shift state to client for non-spoofed

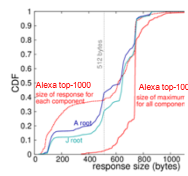
**an amplification attack:**
attacker, **forging IP** of victim
Q: ANY for example.com ?
(~60 bytes)
*server: let me help you, here's 4000 bytes of what I know about example.com*
**result: 60x more bits on victim**



## Leaving Limits

- for >25 years, *policy* decisions forced by UDP packet sizes
  - number of root servers: all fit in 512B
  - DNSsec: required EDNS for >512B
  - crypto algs and key sizes: pkt size limited
  - key rollover: temporary 2x size
- partial fix: EDNS0 deployment (10+ years, since 1999)
- what uses already discarded as too big?

***=> enough already!***




*response sizes today*



## 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 problems
  - *everything* sent in the clear: *no privacy*
  - nothing about DoS
  - large signatures stress UDP size limits

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



## T-DNS: TCP and TLS Connections

- introduction
- why
- **how**
- at minimal cost
- better than alternatives
- next steps

## (Review) Our Contributions

- analysis: **don't fear connections for DNS**
  - client latency: only modestly more
  - server memory: well within current hardware
- implementation choices to get here
- small protocol addition: TLS upgrade



## (Review) Our Contributions

3. analysis: **don't fear connections for DNS**
   - client latency: only modestly more
   - server memory: well within current hardware
2. implementation choices to get here
1. **small protocol addition: TLS upgrade**

*(going in reverse order)*

USC Viterbi T-DNS / DNS-OARC

## Protocol Changes: Goals

- minimize change
- reuse existing approaches
- follow IETF patterns

*(as boring as possible)*

USC Viterbi T-DNS / DNS-OARC

## Protocol Changes: Goals

- minimize change
- reuse existing approaches
- follow IETF patterns

*(as boring as possible)*

- **implications:**
  - **reuse TLS: Transport Layer Security**
  - **add a STARTTLS-like "upgrade"**
  - **innovation: careful implementation**

USC Viterbi T-DNS / DNS-OARC

## 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>

USC Viterbi T-DNS / DNS-OARC

## SMTP *with* 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*

*prologue: in clear (no privacy here)*

**C: STARTTLS**
***S: 220 Go ahead***
**C & S: <negotiate a TLS session with a new session key, in binary>**

*transition to TLS*

**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>**

*contents now private*

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

USC Viterbi

## Our STARTTLS for DNS

(in draft-hzhwm-start-tls-for-dns-01)

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 tfc



## (Review) Our Contributions

3. analysis: **don't fear the DNS connection**
   - client latency: only modestly more
   - server memory: well within current hardware
2. **implementation choices to get here**
1. small protocol addition: TLS upgrade

*(going in reverse order)*



## Careful Implementation Choices

- problem: no tuning of DNS TCP for queries *(until now!)*
- connection reuse (or restart)
  - persistent connections
  - TCP fast open
  - TLS resumption
- query pipelining
- out-of-order processing



## 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

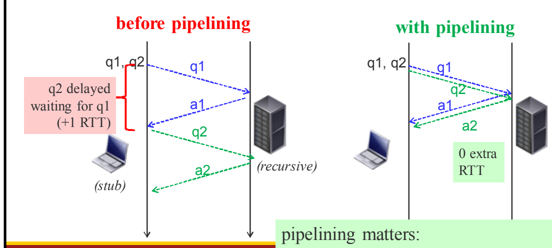


## Connection Reuse

- 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)

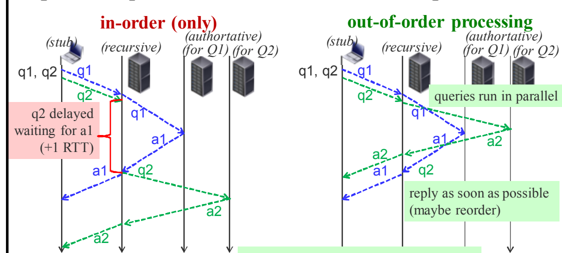



pipelining matters:
62% of web has 4+ domain names
(dataset: common crawl)

USC Viterbi

## Out-of-Order Processing

process queries on same connection in parallel




out-of-order matters:
avoid head-of-line blocking

USC Viterbi

## T-DNS: TCP and TLS Connections

- introduction
- why
- how
- **at minimal cost**
- better than alternatives
- next steps

USC Viterbi T-DNS / DNS-OARC

## (Review) Our Contributions

**3. analysis: don't fear connections for DNS**
- **client latency: only modestly more**
- **server memory: well within current hardware**

2. implementation choices to get here
1. small protocol addition: TLS upgrade

*(going in reverse order)*

USC Viterbi T-DNS / DNS-OARC

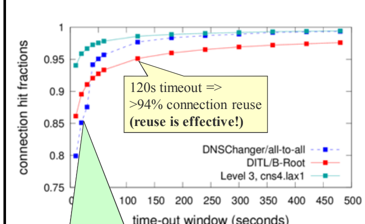
## (Review) Our Contributions

**3. analysis: don't fear connections for DNS**
- **client latency: only modestly more**
- **server memory: well within current hardware**

questions:
- a. connection reuse: hit rate? memory?
- b. CPU cost?
- c. latency:
  - i. stub-recursive?
  - ii. recursive-authoritative?
  - iii. end-to-end?

USC T-DNS / DNS-OARC

## 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)

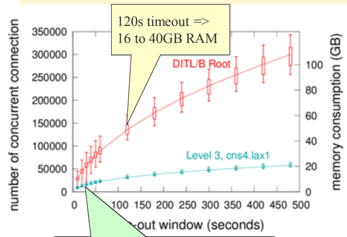
we propose 30s/60s (conservative)
=> still >85% connection reuse

conclusion: connection reuse is *often helpful*

USC Viterbi T-DNS / DNS-OARC

## Cost of Connection Reuse? (ok!)

120s timeout => 16 to 40GB RAM




we propose 30s/60s (conservative) => 9GB for L3, 18 for B-Root

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

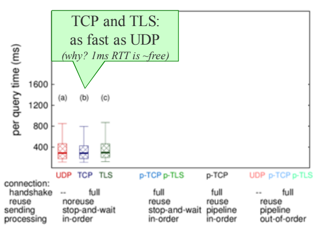
| step | OpenSSL | GnuTLS |
|---|---|---|
| TCP handshake processing | 0.15 ms | |
| TCP packet handling | 0.12 ms | |
| TLS connection establishment | 25.8 ms | 8 ms |
| key exchange | 13.0 ms | 6.5 ms |
| CA validation | 12.8 ms | 1.5 ms |
| TLS connection resumption | 1.2 ms | 1.4 ms |
| DNS resolution (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: as fast as UDP (*why? 1ms RTT is ~free*)




TCP and TLS vs. UDP?
effects of implementation choices?
*with short RTT (1ms)*

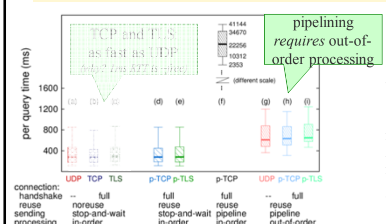
**method**: live experiments of random 140 names from Alexa top 1000; stub-recursive RTT=1ms

(graph shows medians and quartiles)



## Latency: Stub to Recursive

pipelining *requires* out-of-order processing




TCP and TLS vs. UDP?
effects of implementation choices?
*with short RTT (1ms)*

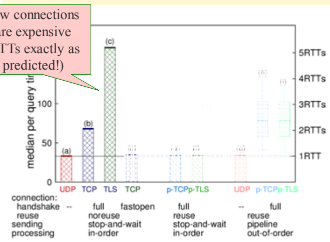
**method**: live experiments of random 140 names from Alexa top 1000; stub-recursive RTT=1ms

(graph shows medians and quartiles)



## 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 repeated 10x; recursive-authoritative RTT=35ms

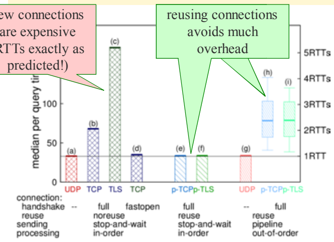
(graph shows medians and quartiles for (h) and (i), or bars where median and quartiles are the same)



## Latency: Recursive to Authoritative

new connections are expensive (RTTs exactly as predicted!)

reusing connections avoids much overhead




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

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

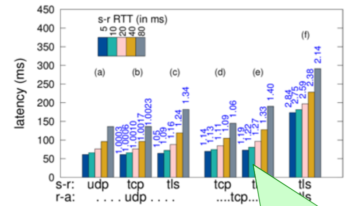
(graph shows medians and quartiles for (h) and (i), or bars where median and quartiles are the same)

## 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, 30s t.o.) + TCP (r-a, 60s t.o.)
19 to 33% slower: **modest cost -> most benefit**



## T-DNS: TCP and TLS Connections

- introduction
- why
- how
- at minimal cost
- **better than alternatives**
- next steps



## Alternatives

- for improving privacy
  - DNScurve/DNScrypt: some neat optimizations to reduce RTTs, but new and fixed stack
  - DNS over DTLS: adds back UDP limits but still stuck with most TLS RTTs
- for anti-DoS
  - on others: rate limiting
- for relaxing limits:
  - seeming alternative: live within UDP limits



## T-DNS: TCP and TLS Connections

- introduction
- why
- how
- at minimal cost
- better than alternatives
- **next steps**



## T-DNS Next Steps

- more information:
  - tech report ISI-TR-2014-688 (www.isi.edu/~johnh/PAPERS/Zhu14a/)
  - internet-draft: draft-hzhwm-start-tls-for-dns-01
- code:
  - client, client & server proxies, unbound patch
  - http://www.isi.edu/ant/software/
- do you want DNS privacy? share feedback?
  - johnh@isi.edu