Using TLS for DNS Privacy in Practice

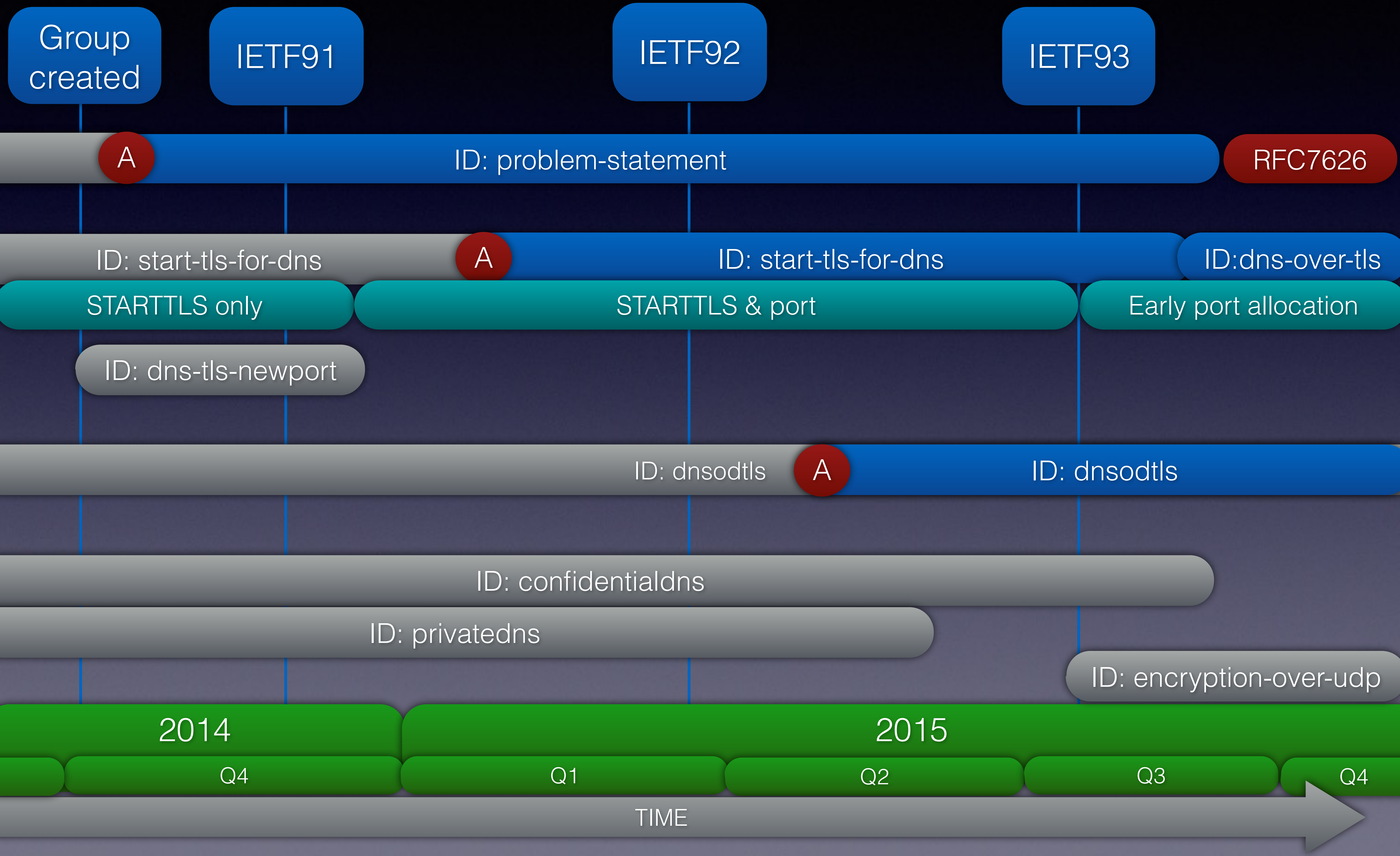
Sara Dickinson
Sinodun

OARC Fall Workshop 2015

Using TLS for DNS Privacy in Practice

- Previous talk - Overview of privacy threats/solutions
- Focus here is
  - IETF efforts (draft-ietf-dprive-dns-over-tls)
  - Implementations - TLS in practice
- Acknowledgements:
  - VerisignLabs
  - NLNetLabs
  - getdns team
  - USC/ISI…

DPRIVE WG



Group created

IETF91

IETF92

IETF93

A

ID: problem-statement

RFC7626

ID: start-tls-for-dns

A

ID: start-tls-for-dns

ID:dns-over-tls

STARTTLS only

STARTTLS & port

Early port allocation

ID: dns-tls-newport

ID: dnsodtls

A

ID: dnsodtls

ID: confidentialdns

ID: privatedns

ID: encryption-over-udp

2014

2015

Q4

Q1

Q2

Q3

Q4

TIME

Pros and Cons

	Pros	Cons
STARTTLS	• Port 53 • Known technique • Incrementation deployment	• Port 53 - middleboxes? • Existing TCP implementations • Downgrade attack on negotiation • Latency from negotiation
TLS (new port)	• New DNS port (no interference with port 53) • Existing implementations	• New port assignment
DTLS	• UDP based • Certain performance aspects	• Truncation of DNS messages (just like UDP) ➡️ Fallback to clear text or TLS ❌Can't be standalone solution • No running code

DNS-over-TCP

- DNS-over-TCP… historically used only as a fallback transport (TC=1, Zone transfer)
- RFC5966 (2010)
  - TCP a **requirement** for DNS implementations
- 2014: USC/ISI paper - Connection-oriented DNS
  - Showed TCP/TLS performance feasible for DNS

draft-ietf-dnsop-5966bis

- Performance (more later):
    - Connection-reuse & pipelining, response re-ordering
- Idle timeouts:
    - Historically clients did “*one-shot TCP* “
      =>(conservative idle timeouts)
    - [edns-tcp-keepalive long-lived DNS-over-TCP sessions]
- Security/Robustness:
    - Historically server implementations were basic in TCP connection management (compare to other protocols)

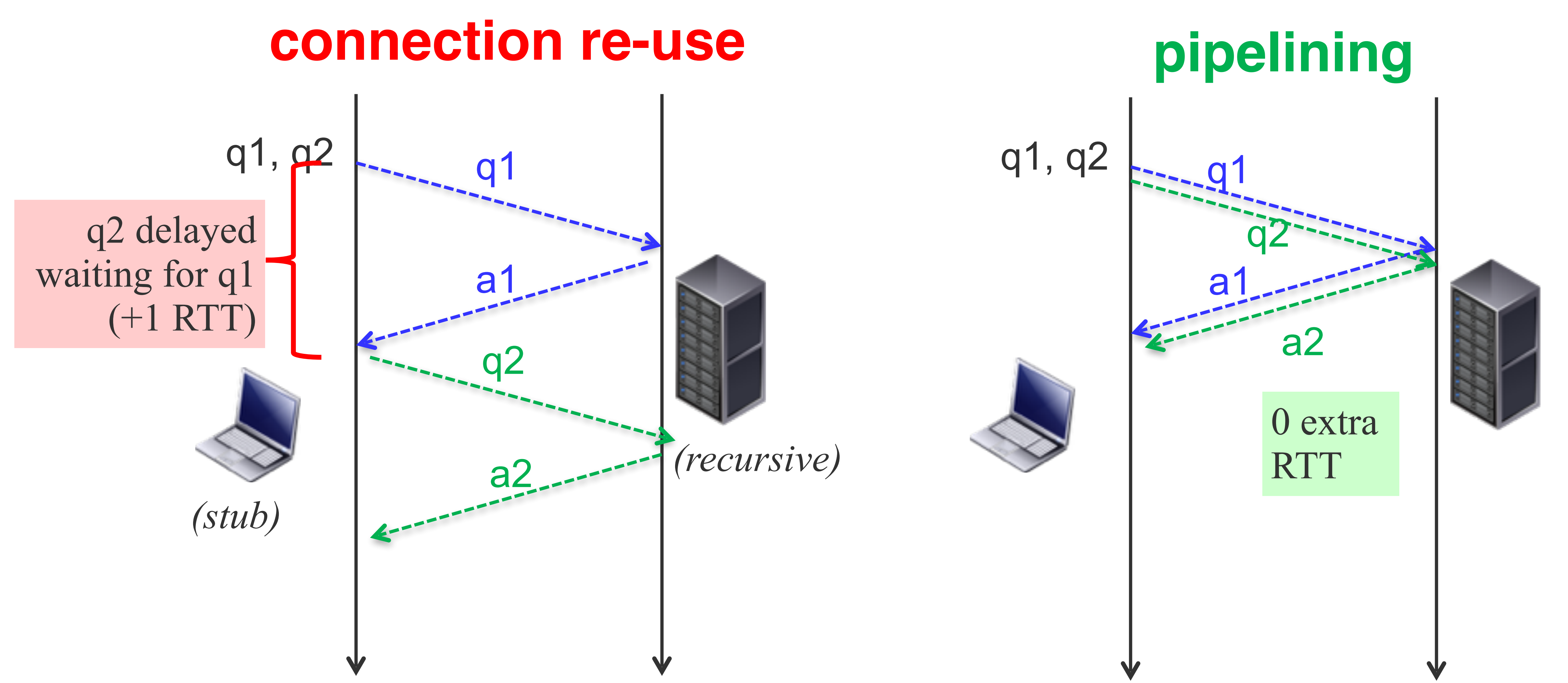
Performance

Goals:

- Amortise cost of TCP/TLS setup
    - Send many messages efficiently
- Optimise TCP/TLS set up & handling

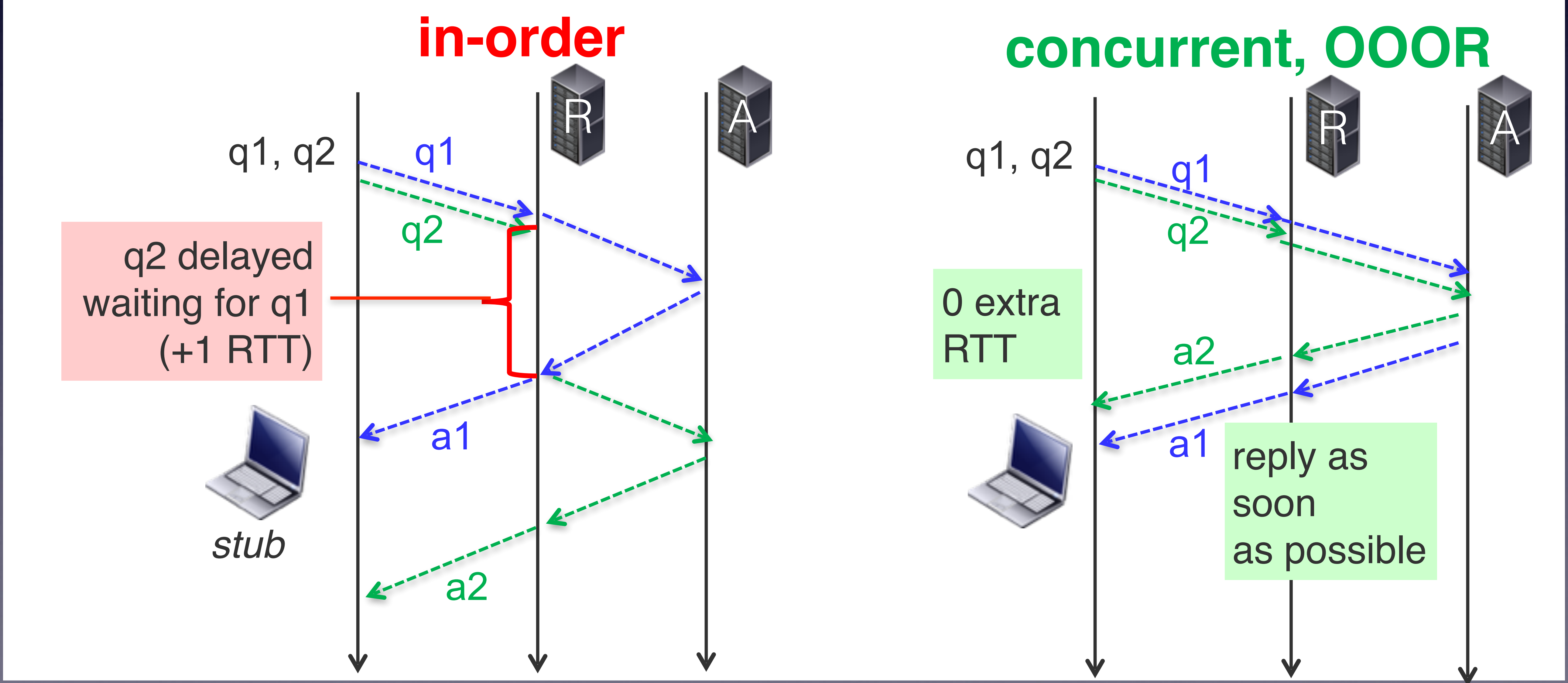
Performance

1. Client - Query pipelining



Performance

2. Server - concurrent processing of requests sending of out of order responses



Performance

3. Reduce latency of **first** message on a connection

- TCP Fast Open (RFC7413)
    - Cookie exchange: Send Data in SYN
    - Linux. [FreeBSD on the way]
- TLS Session Resumption (RFC5077) -Abbreviated handshake using session ticket
- [TLS False Start]

Performance

4. Other considerations (learn from e.g HTTPS):

- Server connection management
  - Robustness and fair use
  - Idle time configuration (keepalive)
- Under the hood - kernel tuning…

TLS in Practice

DNS-over-TLS history

- Production TLS in **Unbound** for DNSTrigger:
  - Unbound 1.4.14 (Dec 2011) !!Pre-dated DPRIVE-WG!!
  - Used to contact open resolver on port 443 (last resort).
- Since then… Prototyping work at **USC/ISI**
- More recently (since 2014)
  - LDNS and NSD TLS patches, BIND TCP improvements
  - **getdns - ongoing development of DNS-over-TLS**

getdns

Unbound security

- Modern **async DNSSEC** enabled API (https://getdnsapi.net/)
- Stub mode has flexible privacy policy
  - **TLS** (port 1021) with transport fallback:
    - Strict (Authenticated) TLS only
    - Opportunistic TLS
    - Fallback to TCP, UDP
- Pipelining, OOOP, Configurable idle time, keepalive (WIP), padding (WIP)

Current status

Software	digit	LDNS	getdns		Unbound		NSD	BIND
mode	client	client (drill)	stub	recursive*	server	client	server	server/ client
TLS	🟩	🟢	🟩	🟩	🟩	🟩	🟢	⬜
TFO	🟩	🟢	🟩	🟢	🟢	🟢	🟢	⬜
Conn reuse	🟩	🟢	🟩	⬜	🟩	⬜	🟩	🟩
Pipelining	🟩	⬜	🟩	🟨	🟩	🟨	🟩	🟩
OOOP	🟩	⬜	🟩	🟨	🟩	🟨	🟨	🟩

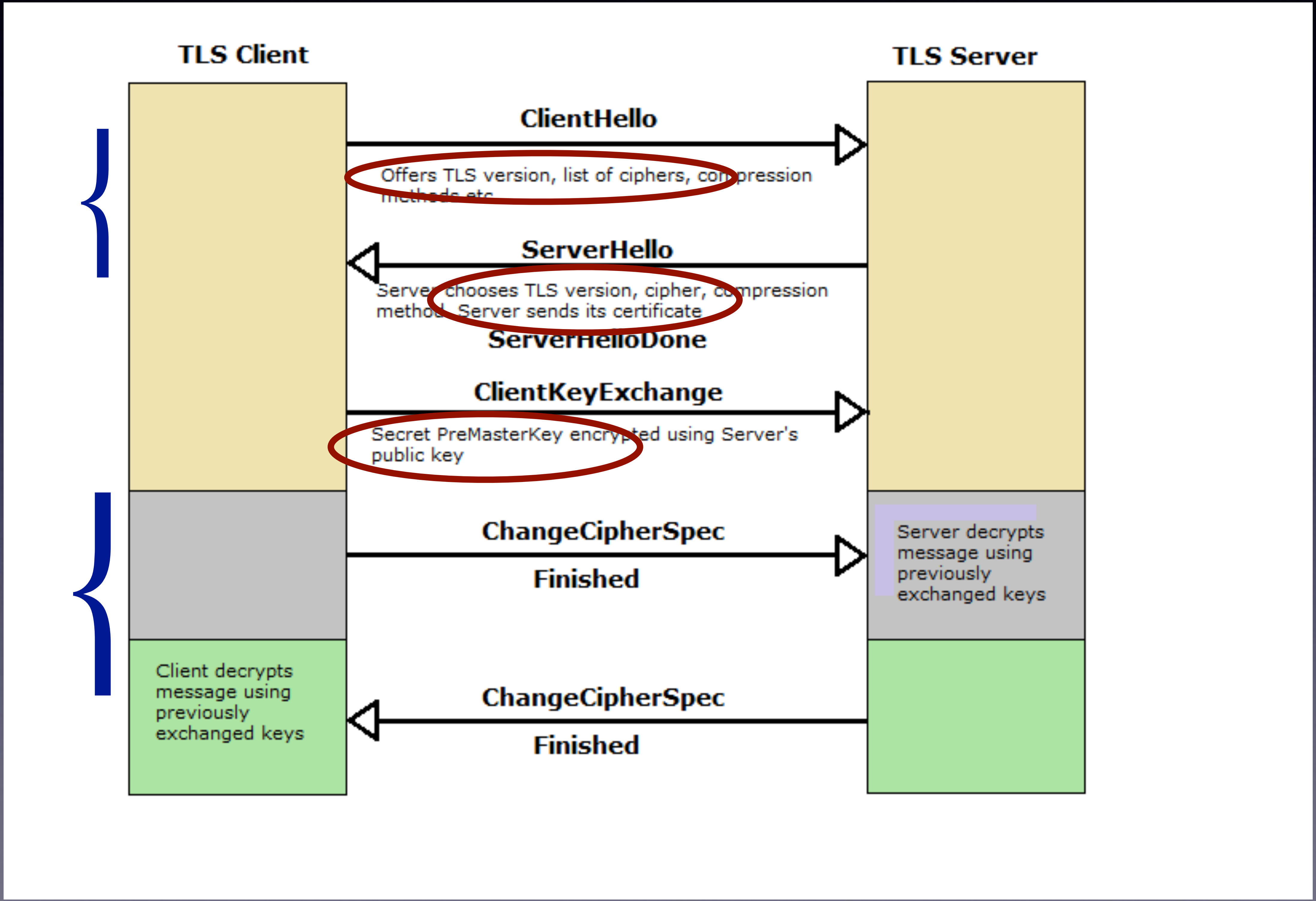
🟩	Dark Green:	Latest stable release supports this
🟢	Light Green:	Patch available
🟨	Yellow:	Patch in progress, or requires building a patched dependancy
⬜	Grey:	Not applicable or not planned

** getdns uses libunbound in recursive mode*

TLS Basics

1 RTT

1 RTT



TLS Client

TLS Server

ClientHello

Offers TLS version, list of ciphers, compression methods etc.

ServerHello

Server chooses TLS version, cipher, compression method. Server sends its certificate

ServerHelloDone

ClientKeyExchange

Secret PreMasterKey encrypted using Server's public key

ChangeCipherSpec

Finished

Server decrypts message using previously exchanged keys

Client decrypts message using previously exchanged keys

ChangeCipherSpec

Finished

TLS BCP

- UTA (Using TLS in Applications) WG produced RFC7525 this year - “BCP for TLS and DTLS”

- Key recommendations - Protocol versions:

But… requires recent OpenSSL

  - **TLS v1.2** MUST be supported and preferred
  - TLS v1.1 and v1.0 SHOULD NOT be used (exception is when higher version not available)
  - SSL MUST NOT be used

DNS-over-TLS is relatively ‘green-field’ - could choose to use 1.2 only

Aside: Cipher Suites

- Cipher Suite specifies combination of:
  - Key exchange/Authentication algorithms
  - Bulk encryption (symmetric) algorithm
  - MAC algorithm
- Recent attacks (RFC7457)
- Forward secrecy for key exchange (DHE)

TLS BCP - Cipher Suites

- Recommended Cipher Suites (4 of ~100):
  - **TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256**
  - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
  - TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
  - TLS_DHE_RSA_WITH_AES_256_GCM_SHA384

AEAD: Authenticated Encryption with Associated Data
(Confidentiality, authenticity and integrity)

TLS BCP - Authentication

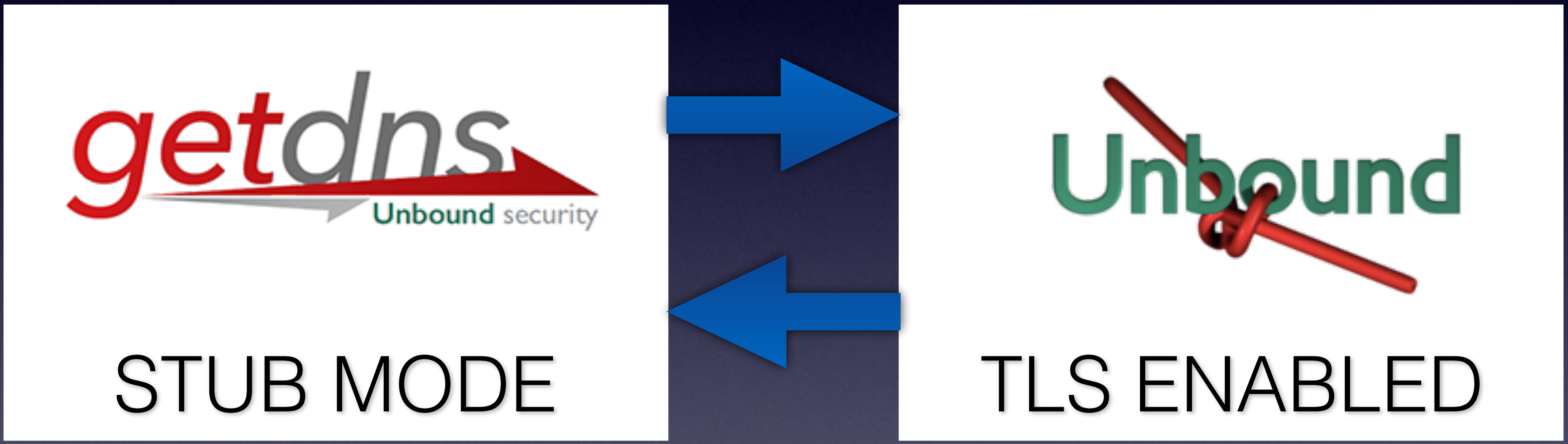
- Secure discovery of certificate/hostname/etc.
  - Indirect or Insecure methods MUST not be used
  - SNI MUST be supported
- DNS-over-TLS
  - Configuration profile
  - DANE (clear-text/un-authenticated TLS, boot strap problem)

But… some view CA’s as weak link

TLS implementations

- OpenSSL
- GnuTLS
- BoringSSL - Google fork of OpenSSL
- LibreSSL - OpenBSD fork of OpenSSL
- WolfSSL, Botan, NSS,….

Examples



STUB MODE

TLS ENABLED

Next release:
Hostname verification

1.5.4

Scenario 1:

Strict TLS

- Configuration:
  - **Hostname verification required (Default)**
  - Correct hostname for Unbound resolver
  - TLS as only transport
- RESULT:
  - TLS used (cert & hostname verified)

Scenario 2:

Strict TLS

- Configuration:
  - Hostname verification required (Default)
  - **No or incorrect hostname**
  - TLS as only transport
- RESULT:
  - Query fails

Scenario 3:

Opportunistic TLS

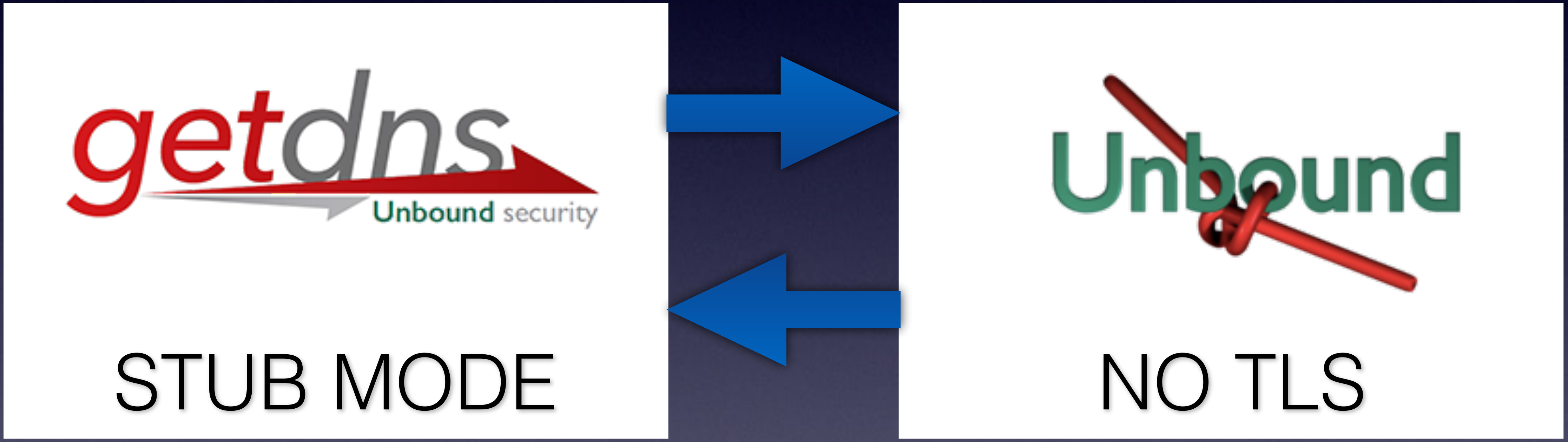
- Configuration:
  - **Hostname verification optional**
  - Valid, none or incorrect hostname
  - TLS as only transport
- RESULT:
  - TLS used (hostname verification tried but fails)

Scenario 4:

Opportunistic TLS

- Configuration:
  - Hostname verification required (default)
  - Valid, none or incorrect hostname
  - **TLS with fallback to TCP**
- RESULT:
  - TLS used (hostname verification tried but fails)

Example



STUB MODE

NO TLS

Scenario 3:

Opportunistic TLS

- Configuration:
  - Hostname verification required (default)
  - Valid, none or incorrect hostname
  - TLS with fallback to TCP
- RESULT:
  - TCP used (TLS tried, but fails)

TLS 1.3

- -08 of draft in progress in TLS WG. Major features:
  - 1-RTT handshake 😃 [0-RTT proposed]
  - Key establishment via (EC)DHE and/or PSK
  - Encryption of handshake messages (e.g. cert)
  - Remove cruft (stricter, simpler implementation)
- Also…TCPINC WG: TCPCrypt vs TLS bun-fight

Summary

- TCP/TLS performance is key
- Consider privacy policy
- Know your TLS BCP
- TLS 1.3 is coming