



DNS Caching: Running on Zero

Randall Atkinson
Extreme Networks



Saleem Bhatti
University of St Andrews



University
of
St Andrews

Domain Name Service

- Distributed name resolution service
- Maps **Fully Qualified Domain Names (FQDNs)** to **DNS records**:
e.g. FQDN (`www.cs.st-andrews.ac.uk`) to a DNS **A record** (IPv4 Address, `138.251.206.45`),
- Also provides other administrative data for specific services (a simple directory service):
 - Name Servers for a domain (NS records)
 - Mail servers (MX records)
 - Jabber (and other) servers (SRV records)
 - Other record types are possible ...

Motivation for examining DNS

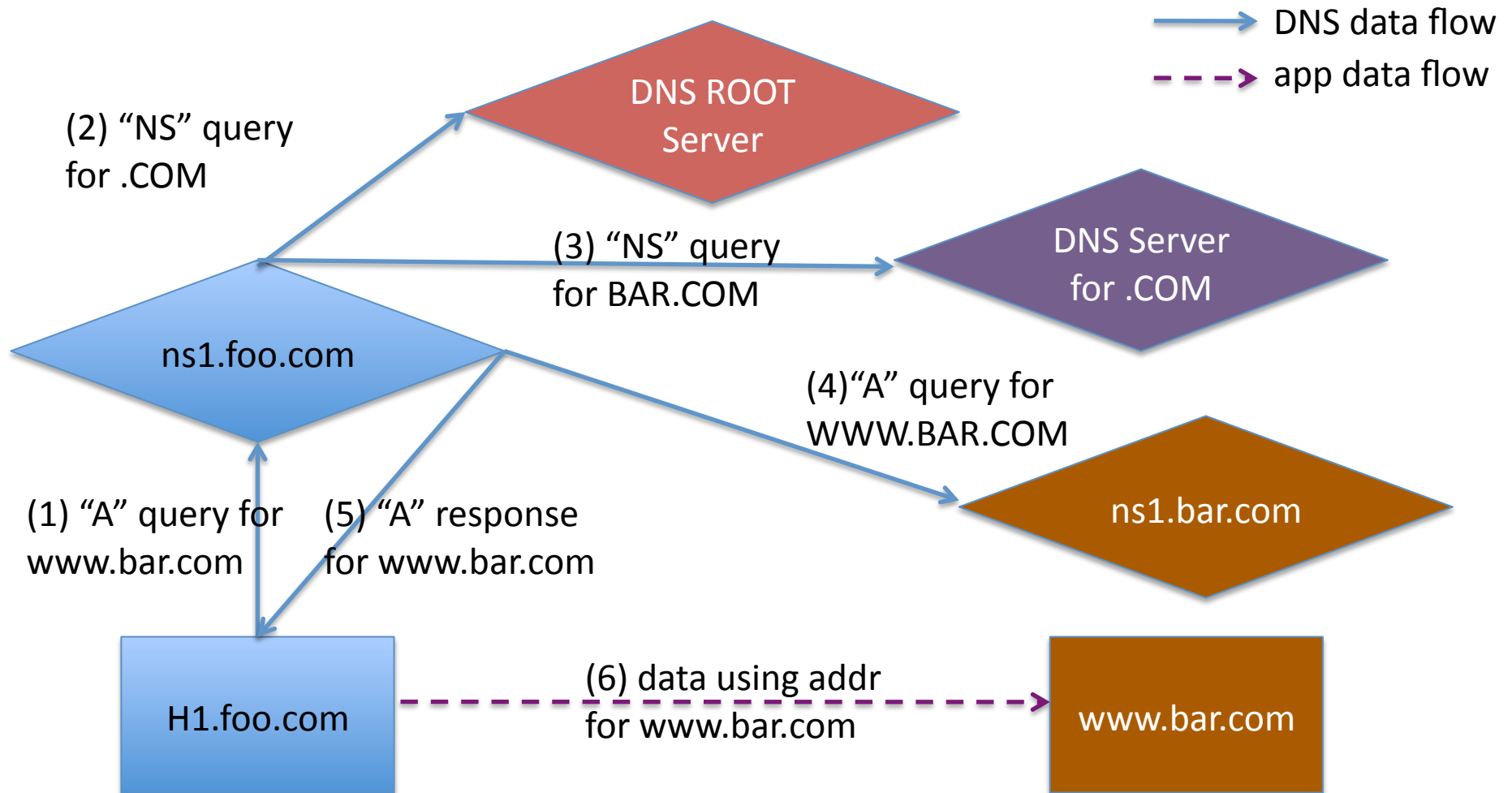
Layer	IP	ILNP
Application	FQDN or IP address	FQDN
Transport	IP address + port no.	Identifier + port no.
Network	IP address	Locator

- ILNP: cleaner semantics for naming in the IP stack:
 - Assumes consistent use of DNS names in applications
 - Uses DNS for mobility, multi-homing, traffic engineering ...
 - <http://ilnp.cs.st-andrews.ac.uk/>
- So, DNS performance is important for ILNP
- But this talk is not about ILNP ...

DNS System Architecture

- Globally distributed **name space**
- Globally distributed **name servers** each holding mappings for part of the name space
- Traditionally, **read-only** for end users
- Enhancements now widely available to enable **write access** for end users:
 - Secure DNS Dynamic Update (RFC-3007)
 - DNS Security (RFC-4033 to 4035) also useful
 - Implemented in BIND, MS Windows, MS Server

DNS Lookup Sequence



Dynamic write access to DNS

- **Write access** is now available for end users
- There is a **temporal caching hierarchy** across the **spatial** distribution of names:
 - Different **records** get cached for different periods of **time**, e.g. NS records and A records
 - Maximum caching time defined by a **Time To Live (TTL)** value held in each DNS record.
- ***Could these two features be exploited in some sensible ways?***

(Non-)Effectiveness of DNS caching

- Jung, J., Sit, E., Balakrishnan, H., and Morris, R. 2002. *DNS performance and the effectiveness of caching*. IEEE/ACM Trans. on Networking. Vol. 10, No. 5 (Oct. 2002), pp. 589-603.
- DNS caching is ineffective for edge sites:
 - **trace-driven emulation** (no experiments)
 - A records could have low TTL (e.g. below 1000s)
 - such low TTL would have low impact on DNS load

DNS experiments at StA [1]

- Experiments in Q1/2008
- Modify TTL values of records in operational DNS server at School of CS, St Andrews
 - 2 DNS servers: BIND 8.2.4 running on Linux
 - ~400 DNS clients: BSD, Linux, Mac, & Windows
- TTL values for successive **7-day periods** during normal semester:
 - Changed DNS TTL on BIND
 - used TTL values **1800s, 60s, 30s**

DNS experiments at StA [2]

- Passive collection of packets via port mirror:
 - *tcpdump(8)* targeting *port 53*
 - Captured all DNS packets
- Results shown on following slides are for:
 - **A record requests** for **servers** only during the capture period (relevant to ILNP, and less ‘noisy’ data)
 - 1 second buckets
- Basic statistics:
 - on time-domain data
- Spectral analysis (**post-pub verification pending!**):
 - examination of request rates
- Analysis: home-brew scripts using NumPy package

Basic dataset meta-data

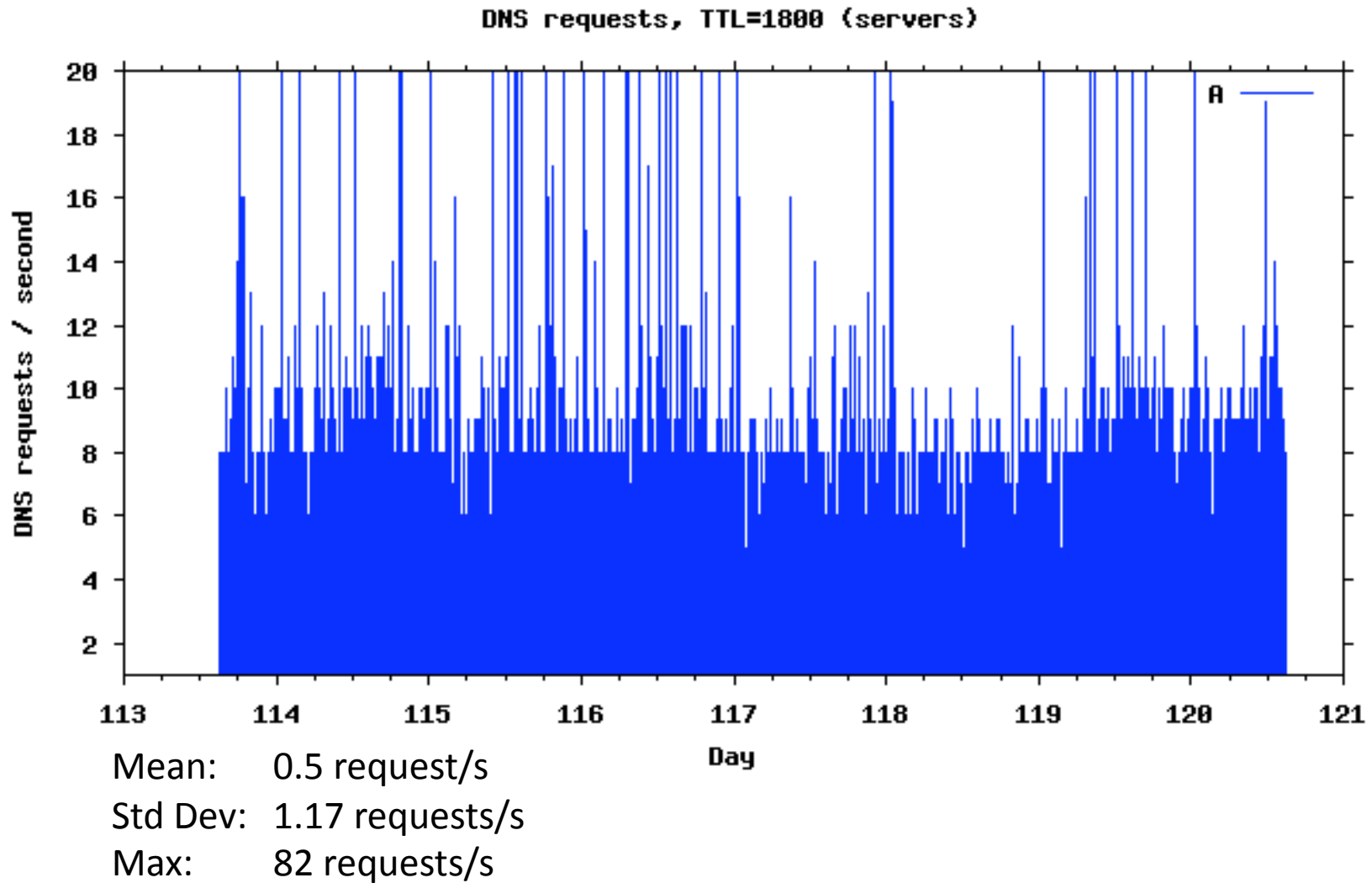
Data set name	TTL [s]	Duration [s] ¹	Total DNS packets captured ²	Number of A record requests for 44 servers ³
dns1800	1800	604,740	9,841,469	303,442
dns60	60	604,739	10,420,760	609,811
dns30	30	604,800	10,979,131	911,537

¹ from tcpdump timestamps, rounded to nearest second, 7 days = 604,800 seconds

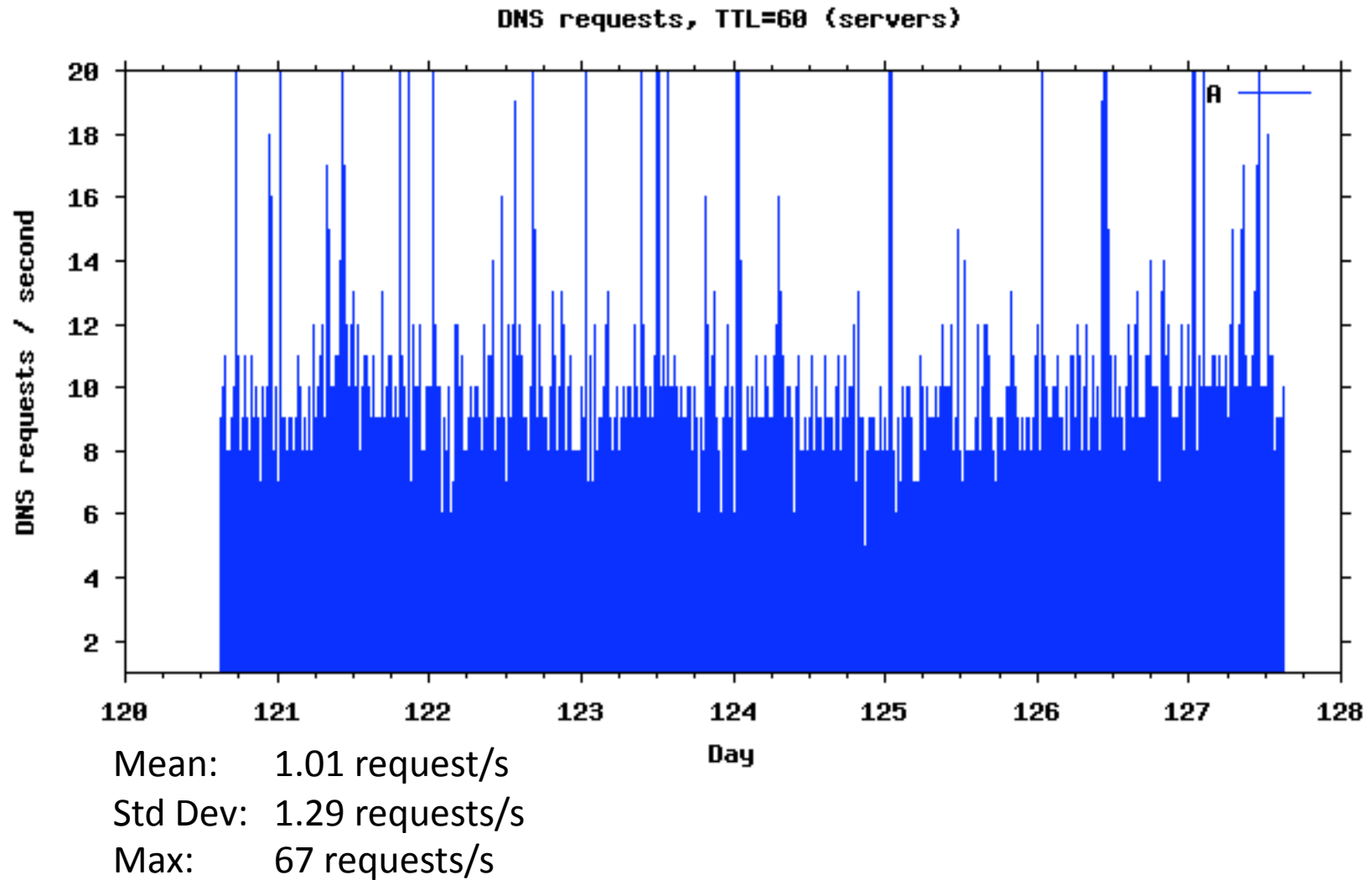
² includes all request and response packets to/from port 53 (TCP and UDP), including erroneous requests etc

³ servers that were active during the 3 weeks of data capture

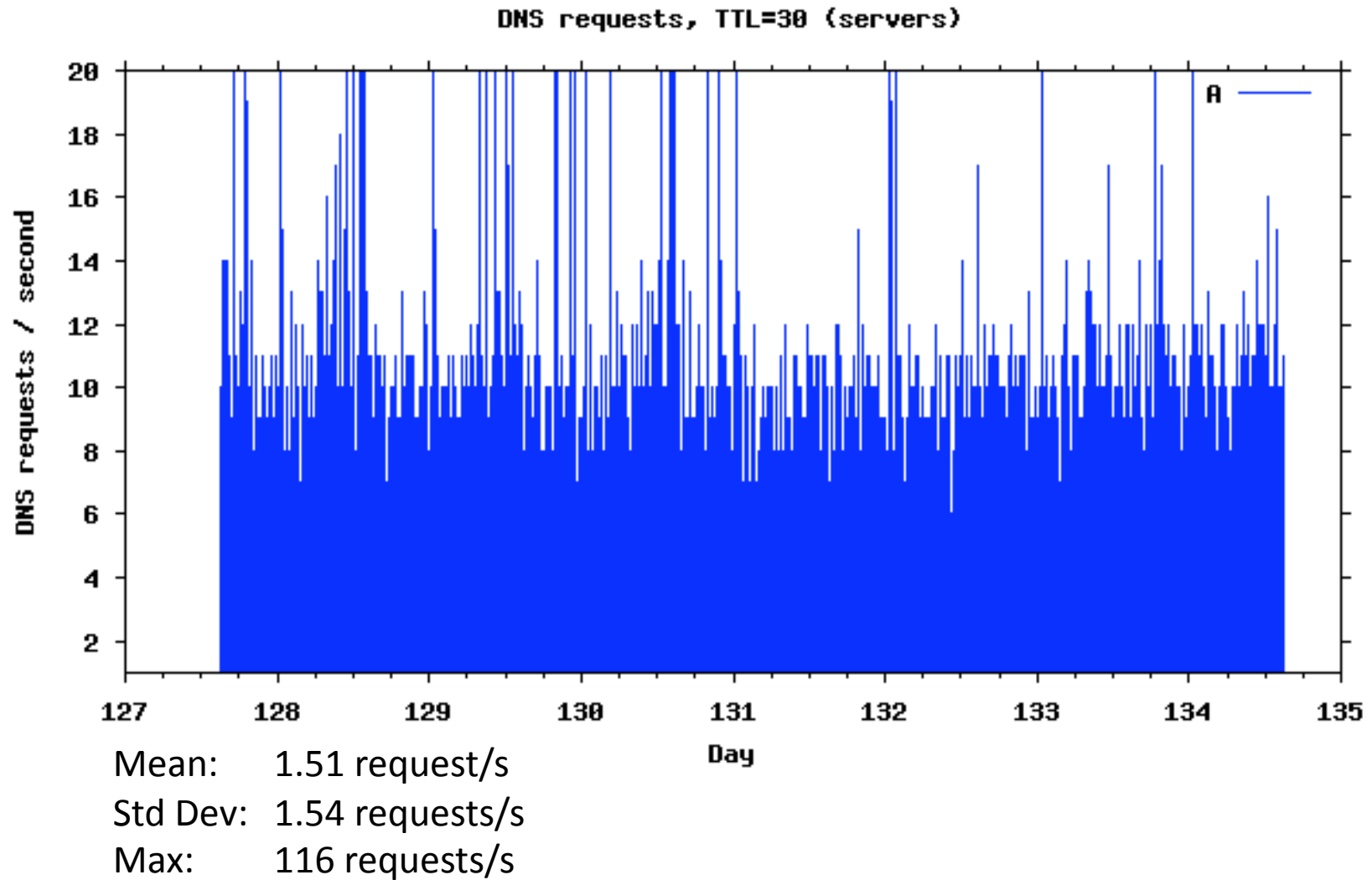
dns1800: A record requests TTL=1800s



dns60: A record requests TTL=60s



dns30: A record requests TTL=30s



Summary of basic statistics

Data set name	Mean [request/s]	Std Dev [requests/s]	Maximum [requests/s]
dns1800	0.50	1.17	82
dns60	1.01	1.29	67
dns30	1.51	1.54	116

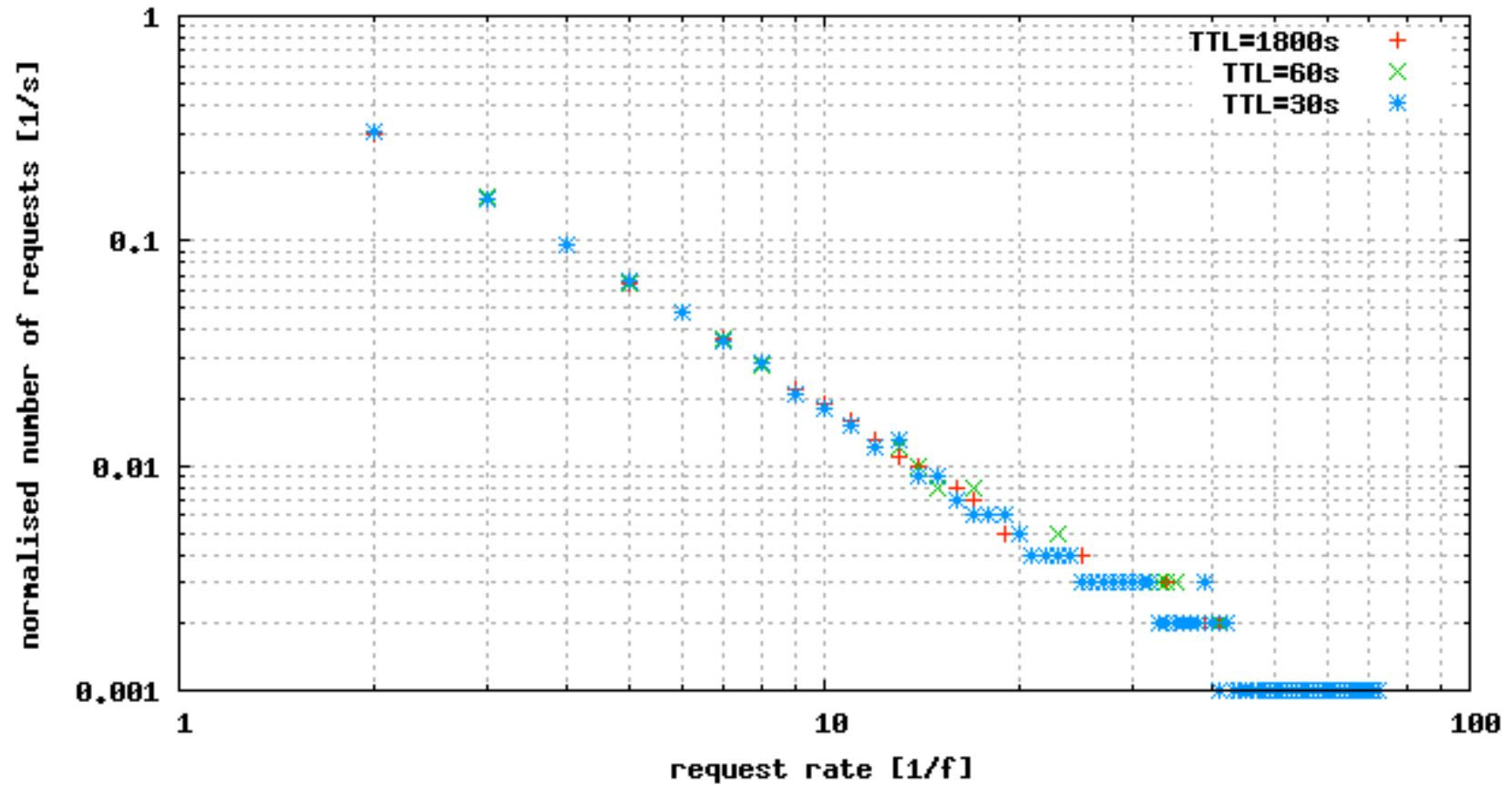
60 fold drop in TTL values
results in (only)
3 fold increase in A record requests

Basic spectral analysis (post-pub verification pending!)

- 1s bucket, sampling at 0.5Hz in the data:
 - assumption about request rates, not verified!
- NumPy:
 - `rfft()` gives real part of FFT (of bucketed data)
 - `fftfreq()` gives frequency components
- Plot request rate using **transformed FFT plot**:
 - y-axis: **normalise** amplitude to show **relative performance** across data sets as **requests/s (1/s)**
 - x-axis: show **request rates** as **1/frequency (1/f)**
- CDF of modified-FFT

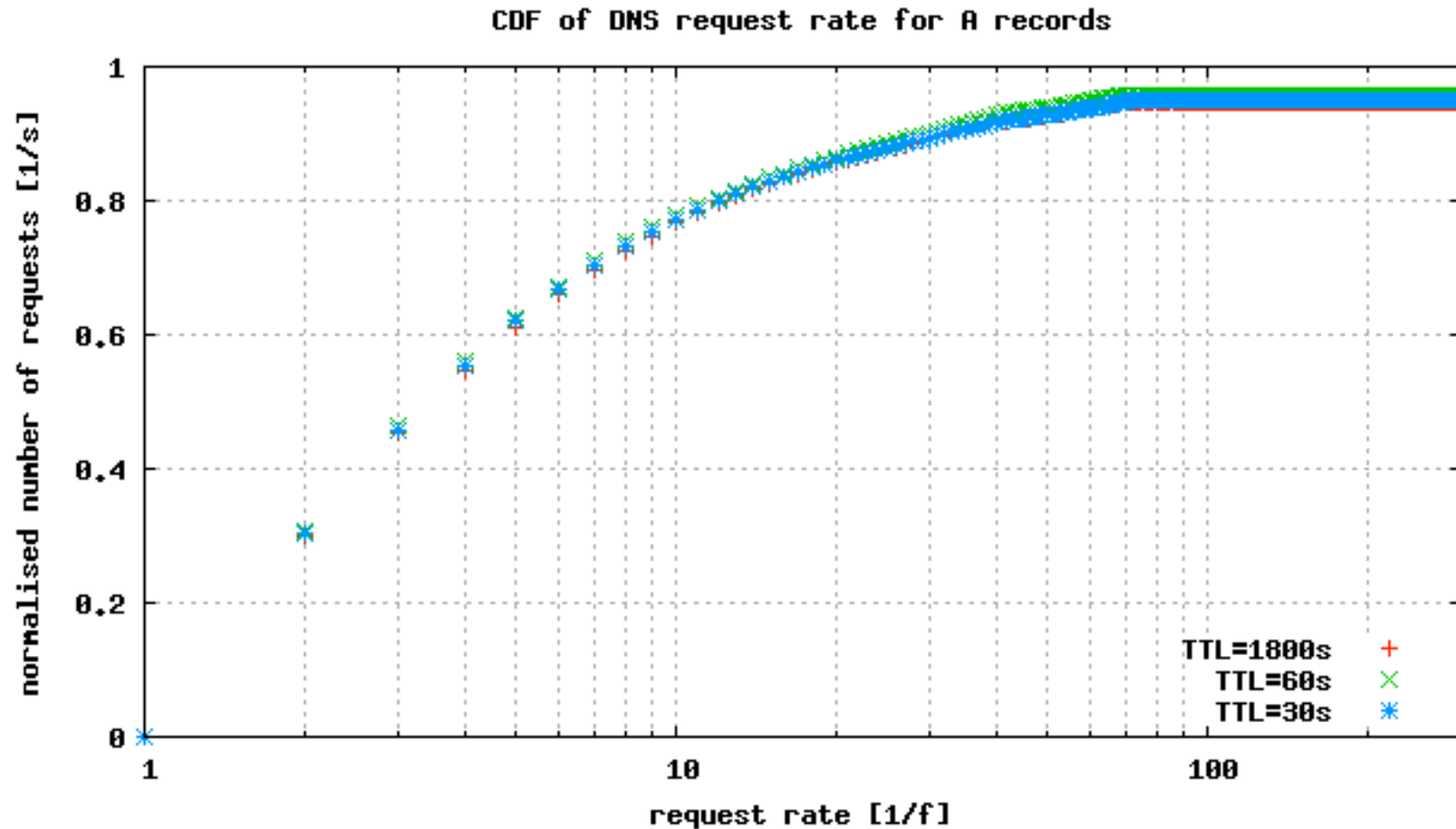
Request rate (from FFT)

DNS request rate for A records (normalised against total number of requests)



Zipf distribution (need to verify).

CDF of request rate



80% of requests < ~13s (need to verify).

Summary of basic spectral analysis

- The overall behaviour of applications in making DNS A record requests for TTL values of 1800s, 60s and 30s is much the same.
- The distribution for DNS A record request rates seem to follow a power-law distribution.
- TTL values for A records below ~ 13 s are likely to be possible without major impact on application behaviour or DNS load.

So far, our discussion presents a technical (academic) curiosity.

The interesting question really is:
“So, what?”

What is possible if DNS TTL were zero?

- Dynamic, frequent, & authenticated DNS updates possible:
 - Simulated by Pappas, Hailes, & Giaffreda, published in LCS 2002
 - Very useful for mobility/multi-homing aspects of ILNP
- High-speed load balancing and VM mgmt for data centres
- Support for mobility and multi-homing:
 - Location updates give changes in connectivity
- Help defend against certain network attacks:
 - DNS cache poisoning for end-sites
 - DDoS: fast-cycle multi-homing (i.e. a kind of “fast-flux” DNS for defence rather than attack)
 - Others possible ...
- Potential for edge-site based multi-path and TE control:
 - multiple Locator values and DNS L record preferences

Who would set DNS TTLs so low?

- Real **A** record values for some servers:
 - TTL = 60 seconds: www.yahoo.com
 - TTL = 20 seconds: content servers in akamai.net
 - TTL = 0 seconds: www.cs.st-andrews.ac.uk
- Note that a site would **NOT** set low TTLs for:
 - Its own **NS** records, which identify its DNS servers.
 - The **A** records related to its NS records.
 - A (mobile) site can remote some or all of its authoritative DNS servers; some sites do so today.

Ongoing DNS experiments at StA

- More measurements in Q1/2009:
 - Site is now using Microsoft Active Directory for its DNS servers
 - Tuning of client-side OS and browser caches to reduce end-system caching effects on DNS data
- Q1/2009 experiments:
 - using TTL values of 1800s, 60s, 30s, 15s, 0s
 - ~150GB of data collected so far
 - analysis in progress ...

Operational Considerations

- Implied semantics of TTL value:
 - **gotcha**: some systems assume that, if network outage time $>$ TTL, then service is down
- PAM2003, PAM2004, NETTS2004 papers by Wessels *et al.*:
 - <http://dns.measurement-factory.com/writings/>
- In fact, the main site has some very interesting reading, including:
 - <http://dns.measurement-factory.com/surveys/200810.html>

Summary and Conclusion

- Summary:
 - Very low TTL values for edge-site DNS records possible
 - DNS load with very low DNS TTLs seems manageable
- Conclusion:
 - Both frequent DNS accesses and frequent Dynamic DNS Updates seem practical to deploy
- What next?
 - More experiments with naming usage (and ILNP)
 - Try experiments with Secure Dynamic DNS updates
- Thanks to:
 - [Martin Bateman](#) for tcpdump/DNS data collection
 - [Systems Group](#) at cs.st-andrews.ac.uk for DNS TTL changes