A Benchmark for HTTP 2.0 Header Compression

Christian Grothoff

Technische Universität München

31.07.2013

https://gnunet.org/httpbenchmark/

Goals

- Realistic benchmark based on diverse, real-world user data
- Preserve realistic privacy expectations of monitored users
- Preserve compression characteristics
- Allow assessment of tunneling multiple requests in one stream

Goals

- Realistic benchmark based on diverse, real-world user data
- Preserve realistic privacy expectations of monitored users
- Preserve compression characteristics
- Allow assessment of tunneling multiple requests in one stream
- Capture data at high-speed link (no TCP stream reconstruction)

Methods

- Capture data with `libpcap` on port 80 that looks like HTTP header
- Traces are headers with same source IP to same destination IP within 5 minutes
- Store in `sqlite` database with trace- and timing information
- Remove IP addresses
- Obscure possibly private data in headers

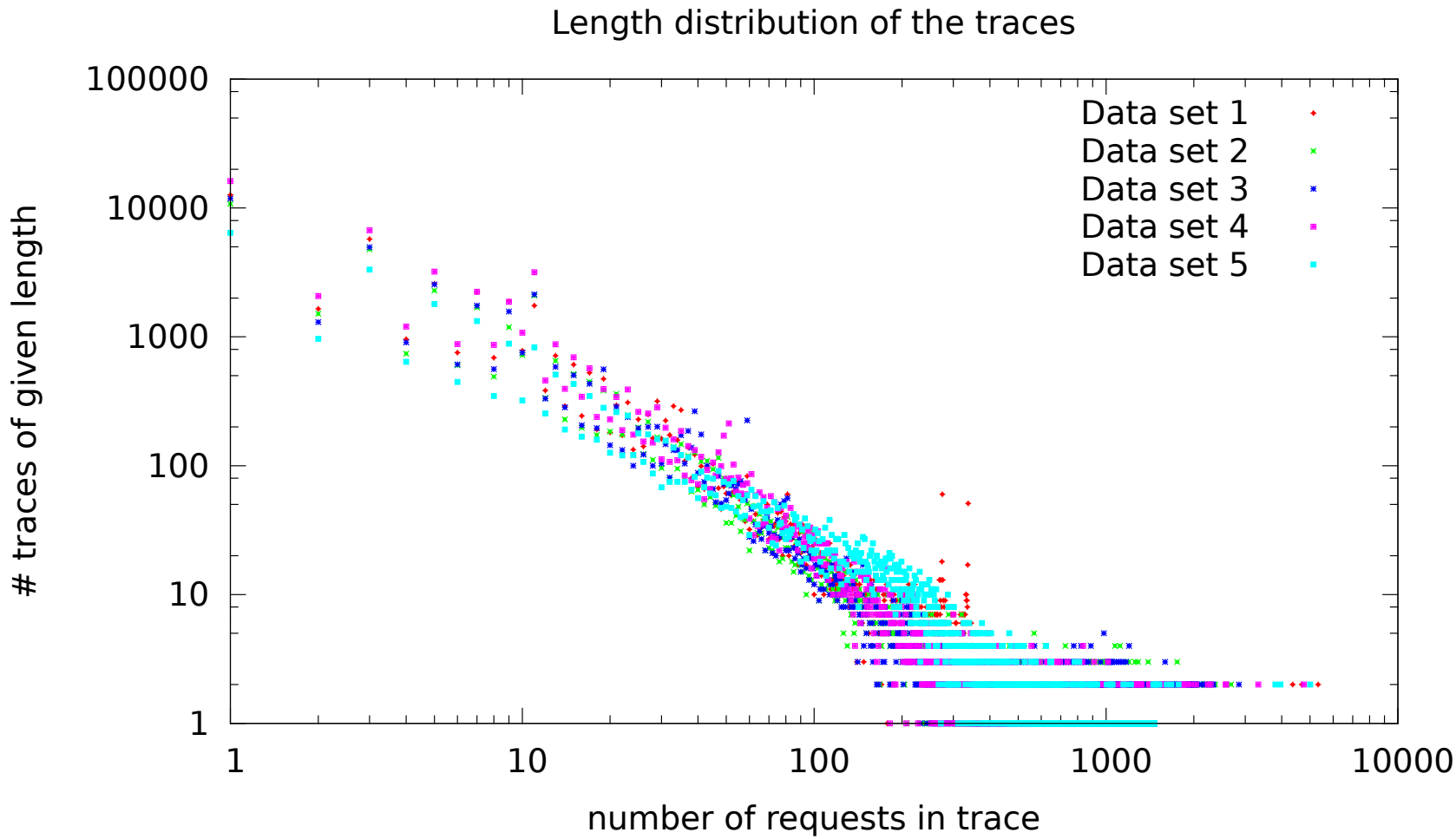
Header cleaning

- Apply substitution ciphers depending on payload:
  - Same substitution for all URIs in a trace
  - Same substitution for all (expected) occurences of hostname
  - Same substitution for cookie values in trace
  - Fresh, character-set preserving substitutions for other headers
- Write regular expressions for headers that are not cleaned
- Discard headers for which we do not have a regular expression

Header cleaning

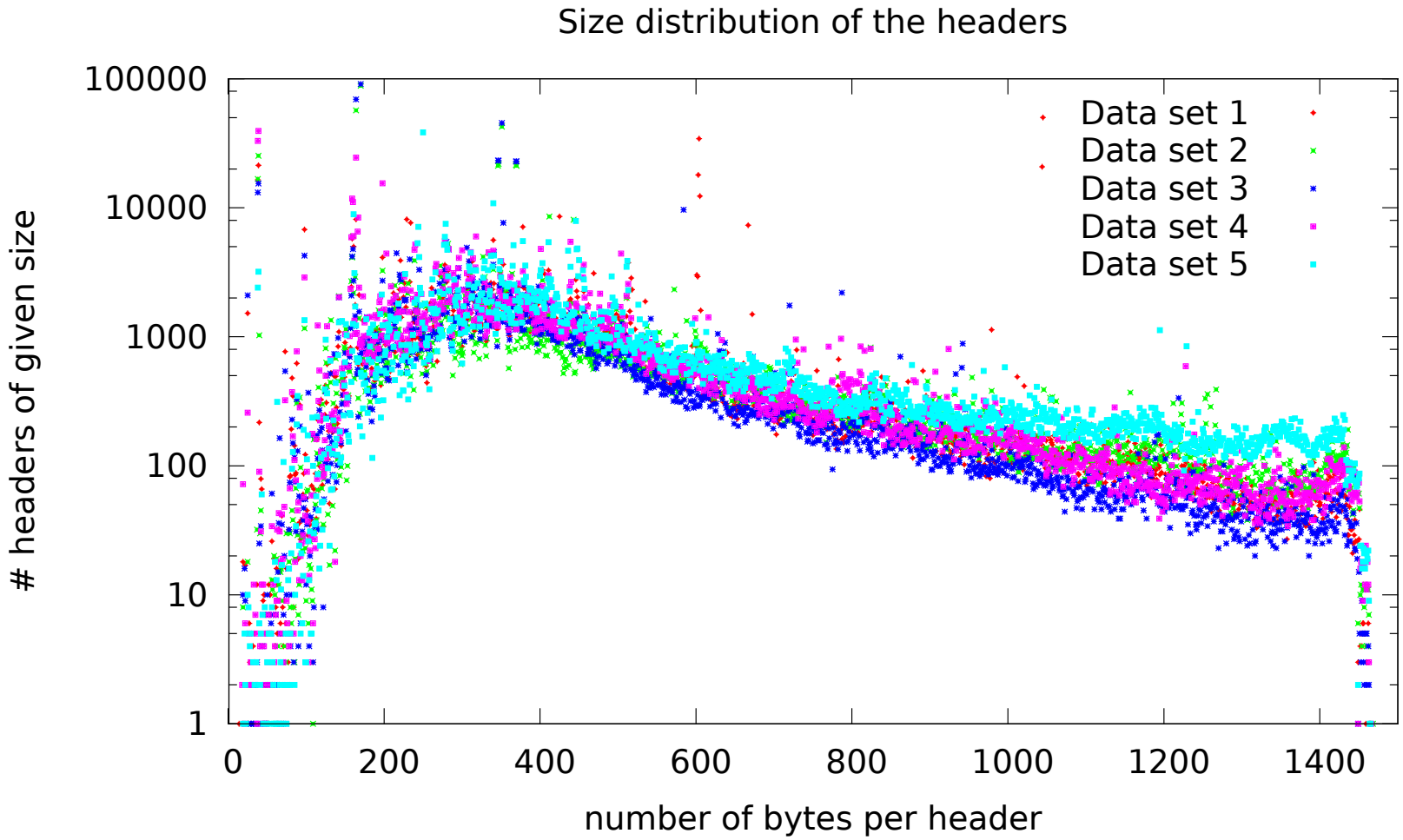
- Apply substitution ciphers depending on payload:
  - Same substitution for all URIs in a trace
  - Same substitution for all (expected) occurences of hostname
  - Same substitution for cookie values in trace
  - Fresh, character-set preserving substitutions for other headers
- Write regular expressions for headers that are not cleaned
- Discard headers for which we do not have a regular expression
- Pass internal review for data export

Length distribution for the traces

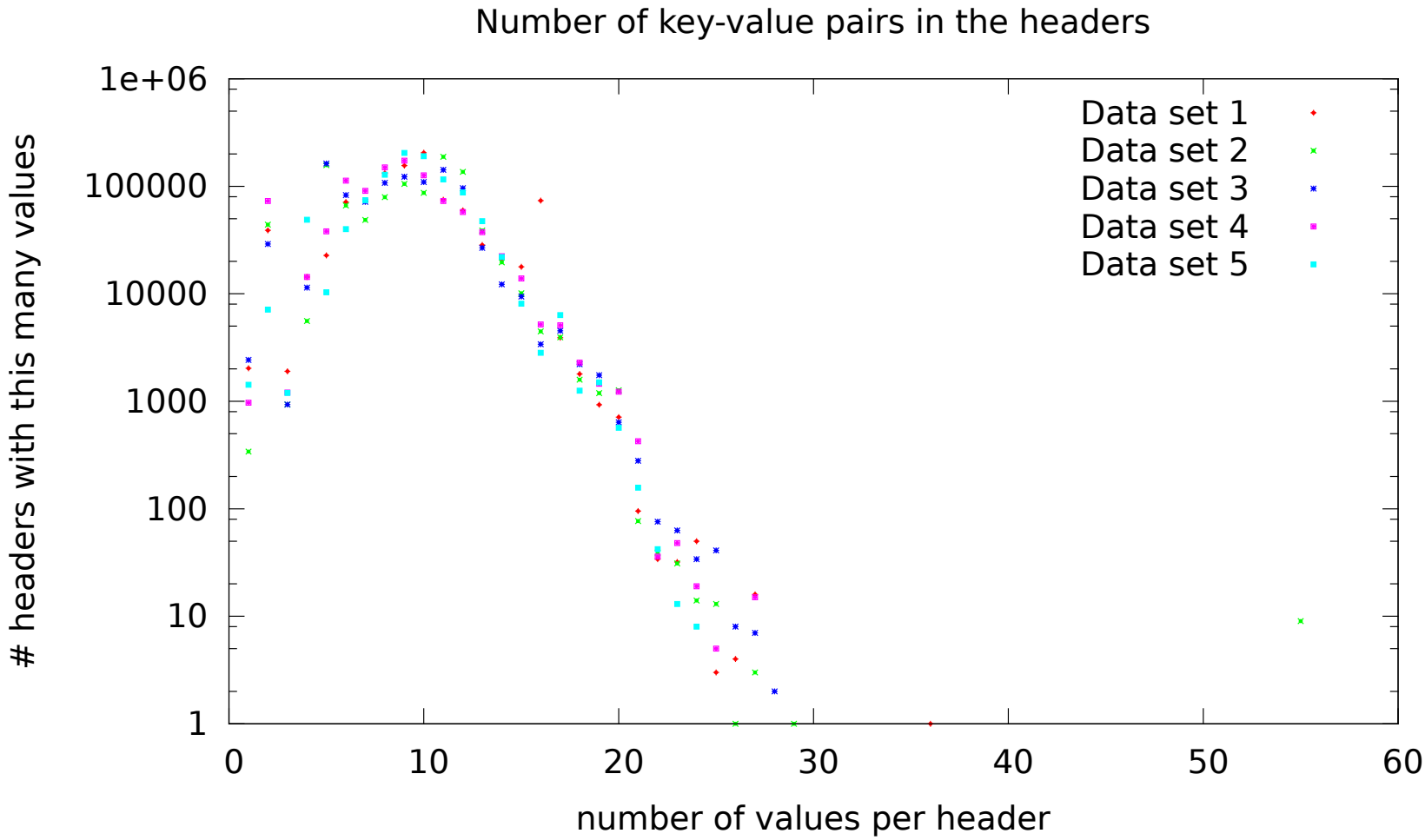


The length of a trace is defined as the number of headers in a given trace.

Number of bytes per header



Number of key-value pairs in the headers



Compression (public benchmark)

Algorithm	Compressed size	Comp. time	Decomp. time
`memcpy` req.	260 ± 44 MB	166 ± 6 ms	148 ± 3 ms
gzip req.	46 ± 18 MB	9,3 ± 2,2s	2 ± 0,5 s
bzip2 req.	215 ± 33 MB	103 ± 14 s	26 ± 4 s
`memcpy` res.	157 ± 12 MB	155 ± 5 ms	137 ± 3 ms
gzip res.	30 ± 4 MB	7,1 ± 0,7s	1,4 ± 0,1 s
bzip2 res.	138 ± 10 MB	70 ± 5 s	17 ± 1,2 s

Compression (raw data)

Algorithm	Compressed size	Comp. time	Decomp. time
`memcpy` req.	265 ± 45 MB	169 ± 4 ms	147 ± 3 ms
gzip req.	37 ± 6 MB	8,6 ± 1,4s	2,1 ± 0,2 s
bzip2 req.	210 ± 31 MB	103 ± 15 s	26 ± 3,9 s
`memcpy` res.	163 ± 12 MB	159 ± 3 ms	138 ± 3 ms
gzip res.	29 ± 4 MB	7,4 ± 0,7s	1,6 ± 0,1 s
bzip2 res.	138 ± 10 MB	71 ± 4,6s	17 ± 1,1 s

Publication

- PDF with detailed description of method
- Five sqlite3 databases with $\approx$ 1 million HTTP headers each
- C source code for capture, cleanup and evaluation
- ODS spreadsheed with statistical analysis of data output by C code
- License: Code is GPL, use of data should be attributed

Do you have any questions?

References:

- Christian Grothoff. *A Benchmark for HTTP 2.0 Header Compression*. `https://gnunet.org/httpbenchmark/`, 2013.

Content-length distribution