Seeing the invisible: two years at Wikipedia with W3C's Network Error Logging

Chris Danis
@cdanis@hachyderm.io
Who's Wikimedia?
Our infra

- All our own metal
- ~1,800 physical servers
- 6 total locations
  - 2x "core" + CDN
  - 4x just CDN
Total request volume 116,867 requests/second

Successful edits 13.8 edits/second

— www.wikimediastatus.net (or https://grafana.wikimedia.org for even more)
“The Internet”
Yes.
<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>timestamp</td>
<td>Oct 28, 2021 @ 19:06:58.574</td>
</tr>
<tr>
<td>age</td>
<td>0</td>
</tr>
<tr>
<td>body.elapsed_time</td>
<td>30,313</td>
</tr>
<tr>
<td>body.method</td>
<td>GET</td>
</tr>
<tr>
<td>url</td>
<td><a href="https://en.wikipedia.org/">https://en.wikipedia.org/</a></td>
</tr>
<tr>
<td>body.phase</td>
<td>connection</td>
</tr>
<tr>
<td>body.protocol</td>
<td>http/1.1</td>
</tr>
<tr>
<td>body.sampling_fraction</td>
<td>0.05</td>
</tr>
<tr>
<td>body.status_code</td>
<td>0</td>
</tr>
<tr>
<td>body.type</td>
<td>tcp.timed_out</td>
</tr>
<tr>
<td>http.client_ip</td>
<td>1xx.2yy.aaa.bbb</td>
</tr>
<tr>
<td>isp_asn</td>
<td>Vodafone Italia DSL (AS 30722)</td>
</tr>
<tr>
<td>http.request_headers.user-agent</td>
<td>Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/94.0.4606.81 Safari/537.36</td>
</tr>
</tbody>
</table>
GET / HTTP/2
Host: en.wikipedia.org

HTTP/2 301
location: https://en.wikipedia.org/wiki/Main_Page
report-to:
  {
    "group": "wm_nel",
    "max_age": 86400,
    "endpoints": [{
      "url": "https://intake-logging.wikimedia.org/..."
    }]
  }

nel: {
  "report_to": "wm_nel",
  "max_age": 86400,
  "failure_fraction": 0.05,
  "success_fraction": 0
}

[...]
<table>
<thead>
<tr>
<th></th>
<th>Chrome</th>
<th>Edge</th>
<th>Firefox</th>
<th>Opera</th>
<th>Safari</th>
<th>Chrome Android</th>
<th>Firefox for Android</th>
<th>Opera Android</th>
<th>Safari on iOS</th>
<th>Samsung Internet</th>
<th>WebView Android</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEL</td>
<td>✓</td>
<td>✓</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>✓</td>
<td>❌</td>
<td>❌</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>71</td>
<td>79</td>
<td>No</td>
<td>58</td>
<td>No</td>
<td>71</td>
<td>No</td>
<td>50</td>
<td>No</td>
<td>10.2</td>
<td>71</td>
</tr>
</tbody>
</table>
Yes.

... 5% of the time
... eventually
Templates/wikimedia.org:intake-logging 600 IN DYNA geoip!text-next

geo-resources:text-next => {  # Returns next-best choice vs text-addrs
  # above, for NEL (T257527 & T261340)
geo-resources-skip_first => true  # This flag is new in v3.3.0
Timeline

- Began work in July 2020
- Launched Sept 2020
- IP geolocation of reports added March 2021
- NEL-based paging starting Oct 2021
Anatomy of a NEL report

- URL
- HTTP method (GET, POST, HEAD, ...)
- Protocol (http/1.1 or http/2)
- Referer
- User-Agent string
Anatomy of a NEL report

- elapsed_time: client-measured duration of the request (until error or success)
- age: how many milliseconds old the report was at the time it was uploaded
- The sampling fraction at generation time
Anatomy of a NEL report

- HTTP response status code, if any
- phase: always one of dns, connection, or application
- type: partial list of examples:
  - dns.name_not_resolved
  - tcp.timed_out
  - tls.cert.date_invalid
  - http.response.redirect_loop
Highest signal

- tcp.timed_out
- tcp.address_unreachable
- dns.name_not_resolved
Noisy signal

- tcp.reset, tcp.closed
- h2.ping_failed
- abandoned
Just noise

- http.error
- unknown
What's not in a NEL

- Client IP address
- Any derived-from-IP data, like a geolocation, AS number, etc.
- Timestamps
Each report is delivered along with an age property, rather than the [local] timestamp at which it was generated. We do this because each user’s local clock will be skewed from the clock on the server by an arbitrary amount. The difference between the time the report was generated and the time it was sent will be stable, regardless of clock skew, and we can avoid the fingerprinting risk of exposing the clock skew via this API.

— https://www.w3.org/TR/reporting/#fingerprinting-clock-skew
Incident case studies: #1
Corrupted PDF downloads

- User reports were our only indication of the problem
- All service metrics looked fine
- Could not reproduce with internal requests
- Inconsistent and intermittent, same file sometimes worked sometimes didn't, no obvious temporality
- At its worst, was affecting about ~20% of downloads
The onion of production

- The corruption was actually truncation
- Initial suspicions were around the PDF generator backend itself, or the reverse proxies nearest to it
The onion of production

- Turned out to be an issue in Varnish (in the middle of our CDN) with a parameter known as nuke_limit

- Full story at https://phabricator.wikimedia.org/T266373
Fix fully deployed approx 13:48
No reports of the PDF truncations in NEL for ~8 hours now, which is a significant break from recent trends. Can anyone else still repro this in any way?

Works for me

— https://phabricator.wikimedia.org/T266373
Incident case studies: #2
Call before you dig

Dear Valued Customer,

Repair crew is still working to expose conduit, they are breaking the concrete and slurry away from around the duct structure ... Techs in the field are reporting that there are no spare ducts available ... There are 5-6 other contractors in the area trying to repair their ducts as well so progress is pretty slow at the moment... No ETR at this time.
Incident case studies: #3
Too much scrubbing

- Use a few different L3/L4 DDoS "scrubbing" services
- "Well, there's your problem"
Otherwise (if result is "Failure"):

1. Increment endpoint's failures.

2. Set endpoint's retry_after to a point in the future which the user agent chooses.

Note: We don’t specify a particular algorithm here, but user agents are encouraged to employ some sort of exponential backoff algorithm which increases the retry period with the number of failures, with the addition of some random jitter to ensure that temporary failures don’t lead to a crush of reports all being retried on the same schedule.

ISSUE 2 Add in a reasonable reference describing a good algorithm. Wikipedia, if nothing else.

---

https://www.w3.org/TR/reporting/
Closing thoughts
Future work @ Wikimedia

- NELs as part of an end-to-end SLO for our CDN
- Mapping real user latency to each of our PoPs using NEL success_fraction
Off the shelf

- At least one CDN now has an "enable NEL" toggle switch

- Several uptime monitoring solutions offer NEL collection
DIY

- Receive a JSON array over HTTP POST
- Do some basic aggregation and analysis, perhaps also IP geolocation
- Set some HTTP response headers in your app
- Have some way of running this outside your usual request path (at least for any given user)
But a few caveats

- Pick your TTLs and sampling fractions wisely
- Reports can be recursive
- Be prepared for background noise — understand it in advance, don't get lost in it
Either way...

- Caveats aside, it's not hard to set up
- Gives you end-to-end observability of real user experiences
- If you run a CDN or use someone else's: Why not NEL?
Further reading & resources

- https://www.wikimedistatus.net/ and https://grafana.wikimedia.org/
- https://www.w3.org/TR/network-error-logging/
- https://www.usenix.org/conference/nsdi20/presentation/burnett