Has REST outlived its usefulness?

Henry S. Thompson
School of Informatics
University of Edinburgh

Slides for a talk given on 21 November 2014 at the Oxford e-Research Centre

Table of Contents

1. Acknowledgements

My years on the Technical Architecture Group of the W3C have stimulated and informed my perspective in many ways.

I'm particularly grateful to Yves Lafon for help interpreting some of the empirical data I'll present today, and to Jonathan Rees for regular reminders to question authority.

2. Before REST: The Web itself

The Web is a the opposite of many objects of study in Computer Science

- Once it got out of Berners-Lee's lab in Geneva and started to grow
- It's always been *ahead* of our understanding of it
- And *way* ahead of the standards that are supposed to describe it

Compare that with a programming language, or a processor architecture, or a protocol

- In the same way that standards *precede* artefacts in the *physical* world

- For most widespread computational artefacts, specification comes first, then implementation

But quite early in its transformation

- From specialised document sharing system for physicists
- To generic information applicance for everyone

The Web inverted that to a surprising extent

3. What is REST: Origins

The HTTP protocol, the URL/URI, MIME and HTML are the fundamental constituents of the Web

- This talk is about the first two

Berners-Lee invented them and implemented them

Their 'official' standardisation first appear in 1994 (URIs) and 1996 (HTTP)

- 4/6 years after its first implementation
- 1/3 years after the first widely available Web browser (Mosaic)

There have been something like 8 or 9 editions/versions of the relevant standards

- IETF RFCs, to be precise

By the second half of the 1990s, the conceptual framework within which the standards were written developed substantially

- And Roy Fielding made major contributions in this area
- Developing what he eventually called Representational State Transfer

As co-editor (with Berners-Lee and others) of the relevant RFCs starting in 1996

- And, in 2000, in his PhD

4. REST: The idea

In his PhD, Fielding sets out to taxonomise distributed information systems

- Or, at least the dimensions along which they might be seen to vary

And identify a particular **architectural style** in terms of that analysis

- Which did two things:
  - Corresponded reasonably well to the (best aspects of the) reality of the Web
  - Explained what made it work so well

An effort subsequently described as

> Seeking to understand the Web so we don't break it by mistake

5. Towards REST: details

Fielding progressively identifies dimensions of variation and picks particular points on each one

Eventually identifying a style of networked information space which is

- **Client-server**
  - That is, takes an asymmetric (request-response) perspective on communication, as opposed to e.g. peer-to-peer
- **Distributed**
  - That is, each server is independent of all others, as opposed to e.g. coordinated or centrally-controlled
- Where interaction is essentially **stateless**
  - > *Each request from client to server must contain all of the information necessary to understand the request, and cannot take advantage of any stored context on the server. Session state is kept entirely on the client.*

> Fielding, R. 2000, *Architectural Styles and the Design of Network-based Software Architectures*, Univ. of California Irvine, available online at http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf

6. Towards REST: the details, cont'd

Client-server interaction should be **layerable** and **cacheable**

- That is, intermediaries are not only allowed, but encouraged

It's important to remember that in the mid 1990s, the majority of home internet usage was only just emerging into the >9600Baud 'future'

- The breakthrough to 28,800 was more or less contemporary with the first release of Netscape Navigator at the end of 1994.

Although bandwidth to large enterprises was better

- The basic unit of bandwidth from telcos was 1.5Mbps
  - As supplied by T1 lines

This was the background to the worldview of standards development throughout the second half of the 1990s

So the *necessity* of caching for decent performance was obvious to all concerned

> *High proxy cache hit rates are* essential *for the web to scale. The architecture has to avoid white-hot servers handling millions of clients directly. Load has to be shifted to local proxies,*

> James Gosling, in a thread with Roy Fielding, Dan Connolly and others, 1995

- And this fit well with another obvious fact
- The **network objects** which were
  - Named (or, indeed, 'addressed') by URIs
  - **Requested** by clients and returned to them in **responses** from servers
- Were static documents
  - Initially, just text
  - The proposal of the `img` tag in 1993 was initially viewed as divisive

7. REST itself

The last part of the original analysis is (obscurely) labelled **uniform interface**

Fielding decomposed this into four constraints:

identification of resources
: That is, the use of *names* (URIs) to put the 'hyper' in hypermedia

manipulation of resources through representations
: Your shopping basket is *represented by* and *manipulated via* web pages, but they are not the basket itself

selfdescriptive messages
: That is, MIME

hypermedia as the engine of application state
: The affordances manifested by the web page are all there is (**HATEOAS**)

Fielding coined the phrase "Representational State Transfer" (**REST**) for this collection of constraints

- *Along with all the other selections made on all the other dimensions*

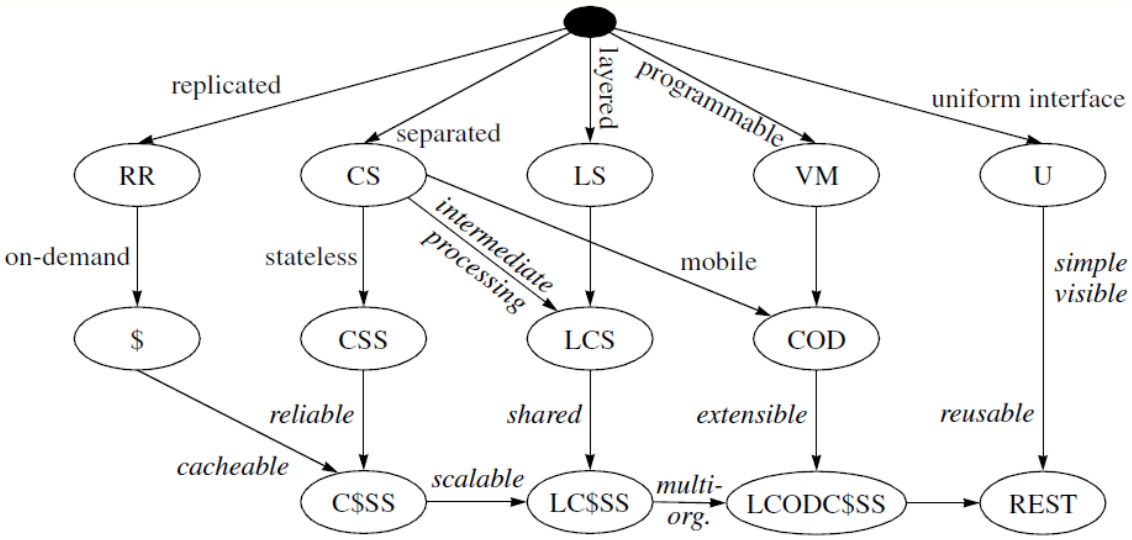


Figure 5-9. REST Derivation by Style Constraints

Fielding, R. 2000, *op cit*.

Observing the constraints [should] guarantee emergent [good] properties such as scalability, robustness, efficiency

8. REST in Fielding's own words

> "Identifiers, methods, and media types are orthogonal concerns" Fielding blog comment

> "the REST architectural style [depends] on the notion that hypertext is a constraint[.]" Fielding blog

> "When I say **hypertext**, I mean the simultaneous presentation of information and controls such that the information becomes the affordance through which the user (or automaton) obtains choices and selects actions." Fielding blog comment

9. What is REST today, in practice?

In the early part of this century, the label **RESTful** appeared

- Descriptive not of the Web as a whole
- But of individual Web-based applications

Focussing in on client-server interactions and, in particular, HTTP

- Even more narrowly than in Fielding's use of the word

Two main slogans, if you will:

Respect safety and idempotence
: That is, `PUT` should be repeatable, and `GET` (and `HEAD`) should be side-effect free
    - For some value of 'side-effect'

Use names
: That is, (*contra* SOAP and XML-RPC), don't `POST` *messages* to get information, **GET** from *URIs*

You can even collapse the core of these two into one:

- GET should be side-effect free, and anything this is side-effect free should use GET

Rather a lot of virtual ink was spilled arguing around this

- But it's not my main interest today
- I want to go deeper into the foundational assertions and assumptions

10. From "network objects" to "resources"

The terminology of the HTTP and URI specs changed over time

- In particular, the name for what it was that the Web consisted of changed
  - From "network object" (concrete)
  - To "resource" (abstract)

And as the focus shifted from a hypertext/navigation perspective

- To a request/response perspective

This necessitated a distinction between what was *connected* (resources)

- And what was sent in messages (representations)

The abstraction was necessary, and indeed fundamental to the value of the Web

- The value proposition of `http://www.guardian.co.uk/` is that it identifies *today's* news

11. The problem, part 1

The rhetoric of the resource/representation distinction has found its way into the standards

- Where it's not only not needed
- But counter-productive, even wrong

HTTP is a network protocol

- And as far as HTTP is concerned, URIs are just protocol elements

Their specification doesn't *need* to define what they're *for*

- And a partial, imperfect definition is worse than none at all

As Jonathan Rees has pointed out

- Underspecification creates extensibility points
- Which in turn may (have) led to interoperability problems

12. The problem, part 2

Caching and content negotiation are just not relevant anymore

Quantitative changes in its technological substrate

- Now require a qualitative shift in our understanding of the Web

Bandwidth is cheap now, caches are expensive

Servers don't just ship out static documents anymore

- And the user agents on clients don't map message bodies to the screen in large monolithic chunks
- (Rendering e.g. the Guardian home page may involve 50–100 HTTP requests)

What you see is not what I see

- Or what I saw a few minutes ago

Virtually all of the main points of Fielding's architectural analysis are now wrong

13. Empirical evidence

The University of Edinburgh runs a caching proxy at the 'edge' of its intranet

- Every HTTP request/response exchange from all the computers in student labs, the library, etc.
    - As well as those from most administrative and support machines
    - And academic / research machines which opt in
- Go through one of four machines running Squid caching software

These caches log information about every request/response pair

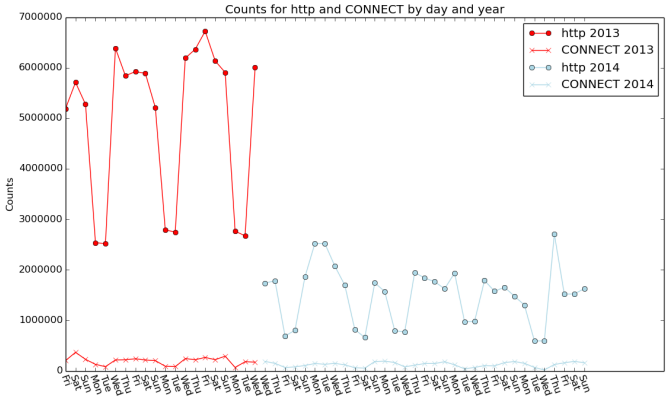
- Including the source and destination, the HTTP request method and result status code and length and whether or not the result was found in the cache

I agreed an approach to anonymising this data with Information Services

- And got the logs for 20 days in June/July of 2013
    - A little over 100 million entries
- And 33 days in June/July of 2014
    - A little over 66 million entries

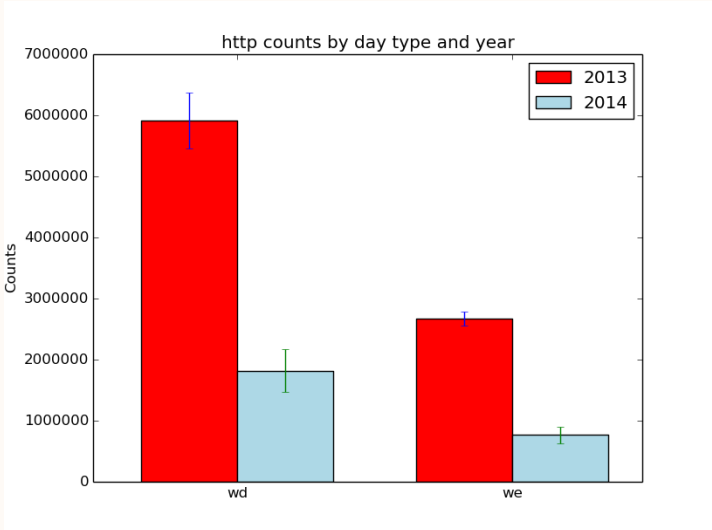
14. The data

Here's a rough idea of what the traffic looks like:

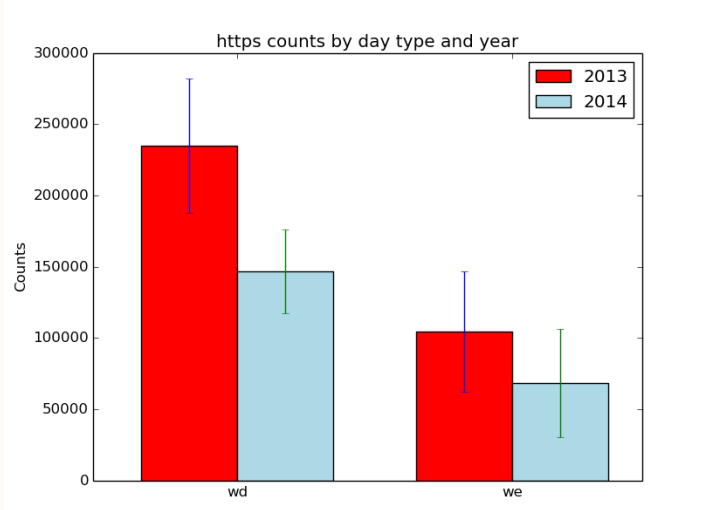


We can compare the daily and weekly average traffic from year to year

- For http



- And the initiation (`CONNECT` request) for https



Two things are evident here:

- The caches are getting less use
- Encrypted traffic is dropping less than in-the-clear traffic

And cache effectiveness is low (and dropping):

- 25.5% of requests were filled from the cache in 2013
- But only 19.8% in 2014

By volume it's much worse overall, and dropping nearly as fast:

- 11.1% of 4TB in 2013
- 9.3% of 2TB in 2014

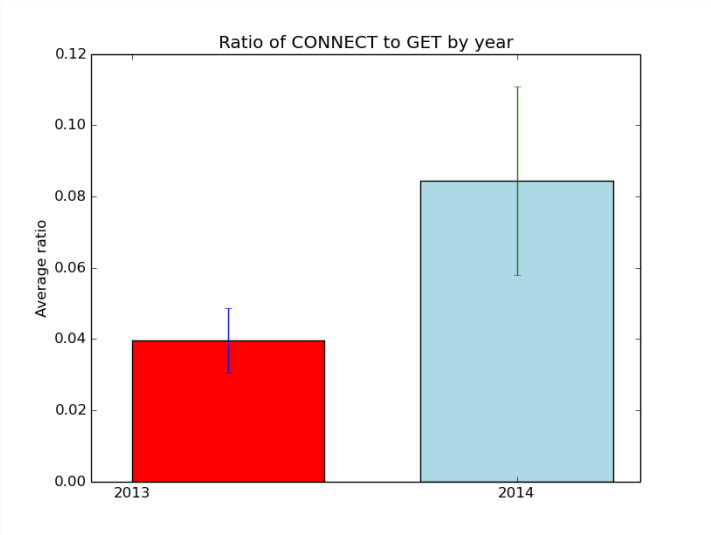
15. The rise of encrypted traffic

The percentage of traffic going via HTTPS is impossible to determine from the Squid logs

- Only `CONNECT` requests are logged
- The actually encrypted transfers are off the books

But we can see some interesting trends

- The ratio of encrypted to in-the-clear is going *up*



And the use of brute-force multiplexing is dropping rapidly

- For example, the proportion of `map1--map9.google.*` wrt to all in-the-clear traffic to `google` drops from 14% in 2013 to 4% in 2014

It's tempting to see this as the impact of the use of SPDY for all Chrome->Google traffic

- Firefox too, I think

16. Content negotiation

There's little evidence that the distinction between reactive and proactive negotiation is understood or exploited

- Despite RFC 2616's recommendation that reactive negotiation is to be preferred
- I can't find any evidence that it is being used *at all*
- In the 2013 logs, out of approximately 76 million `http:` requests, the responses relevant to reactive conneg occurred as follows:
- code n
  300 109
  406 52

Subsequent exploration of the URIs involved determined that *none* of the `300` were the result of a server attempting to offer reactive negotiation

- They were all coming from Apache's `mod_speling` module
- "common basename", "character missing" or "mistyped character"

The `406` cases were harder to be sure about

- Only 13 distinct URIs were involved
- With mostly 0-length response bodies

17. Summary of problems

Where does this leave us?

- Caching (as originally understood) is no longer relevant
- Content-negotiation is of marginal significance at best
- The resource-representation distinction creates more problems than it solves
- The dual nature of URIs as protocol elements and URIs as constitutive of a global namespace is neither well-defined nor well-understood

18. Conclusions

To answer the question in my title:

- No, REST itself, narrowly understood, is still a useful model for (some) Web applications
- Respecting the idempotence of GET, and using POST only when you can't use GET, has clear benefits

But

- Many of the REST architectural style's choices along other dimensions have been overtaken by events
- We need to rethink our stories about the Web and how it works

In particular, we need to

- Look carefully at the possibility of layering our account of HTTP better
- bring *presentation* and *people* (back) into the story

Without an account of how the Web is *used*, we can't understand what it *is*