# HTTP Status Report

Mark Nottingham

IETF HTTPbis WG Chair <mnot@mnot.net>
Principal Technical Yahoo! <mnot@yahoo-inc.com>

# Hidden Agenda

- Inform what HTTP (the protocol) can do
- Inform what implementations can't (yet) do
- Encourage implementers to close the gap

# Status of the Standards

# HTTP circa 1996

- HTTP/0.9 fading quickly
- HTTP/1.0 taking off
- HTTP/1.1 to contain the damage
  - virtual hosting
  - persistent connections
  - caching
- HTTP-NG discussions already underway

# HTTP circa 1996

- Typical use
  - Browser client, static or CGI content
  - GET, POST
- WebDAV: Glimmer in Whitehead's eye
- Services: huh?

# 2002: BCP56

- "On the use of HTTP as a Substrate"
- Reasonable advice for the IETF community, but failed to foresee "services" and "Web 2.0"
- Codified distaste with non-browser uses
  - A new port for every app
  - Probably a new URI scheme too
- Currently being considered for deprecation

# HTTP in 2008

- HTTP/2.0 didn't happen
- WS-* debacle unfortunately did
  - PEP turned into SOAP
- "RESTful" APIs
  - HTTP as Protocol Construction Toolkit
  - Big surprise: Atompub
- Pressure to extend
- Explosion of implementations
  - new servers, clients
  - new frameworks, APIs

# HTTPbis

- IETF Working Group to
  - incorporate errata
  - clarify ambiguities
  - document extensibility
  - improve interoperability
- I.e., writing the recipe down more clearly
- Specifications need to outlive their creators
- NOT to extend HTTP (but wait...)

# HTTPbis: specs

- Problem: RFC2616 is 176 pages of text/plain
- Solution: split it up
  - p1: messaging
  - p2: semantics
  - p3: payload
  - p4: conditional requests
  - p5: ranges
  - p6: caching
  - p7: authentication

# HTTPbis: fixing...

- Currently 139 issues, like
  - ABNF conversion
  - Whitespace between header name and colon
  - Registries for status codes, methods...
  - Vary corner cases
  - Clarify handling of bodies on GET requests
  - Header i18n and folding
  - ETags on PUT responses
  - Get rid of 305 Use Proxy
  - Clarifying the cache key

# HTTPbis Status

- Currently on draft -05
- Major editorial rewrites starting
  - p1 messaging
  - p5 caching
- After that, most should be downhill
- “six months”

# Status of the Implementations

# Implementations

- **Clients**
  - IE, Mozilla, Opera, Safari, wget, curl, serf, Perl, Python, Ruby, Java
  - Abstractions: XmlHttpRequest, Prototype.js, Flash APIs
- **Servers**
  - Apache, IIS, Lighttpd, your router, phone and fridge
  - Abstractions: filesystems, CGI, WSGI, Servlet
- **Intermediaries**
  - Squid, Network Appliance, ISA, HAProxy, tinyproxy, load balancers, firewalls
  - Not many abstractions (yet)
  - 20%-30% of Web traffic goes through a proxy
- **Caches** in clients and intermediaries
  - starting to show up in Python, Ruby...

# HTTP Versions

- Most everything these days is HTTP/1.1, except...
  - Squid (full 1.1 coming)
  - wget
  - a few libraries
  - very old browsers, servers, libraries
- That's OK

# Core Methods

- GET, POST - universally supported
- PUT, DELETE
  - A *few* clients can't generate (e.g., Safari2 XHR)
  - Intermediaries can be configured to block, but usually aren't (except the paranoid and mobile)
- Biggest limitation is W3C languages
  - XSLT, HTML forms
- Result: X-HTTP-Method header (Google) or query params (e.g., ?real-method=POST)

# “Advanced” Methods

- OPTIONS
  - Hard to configure in servers
  - Isn’t cacheable... oops.
  - Result: only used for esoteric protocols (*dav)
- Extension methods - FOO
  - A number of clients don’t allow (e.g., XHR)
  - Intermediaries often block (e.g., Squid, L4 balancers)
  - Result: This probably isn’t so horrible

# URIs

- Mobile clients limit to as small as 256
- Browsers
  - IE: ~2k
  - The rest: really really big
- Intermediaries are OK up to about 4k; some go higher
- Servers can be configured (or replaced)
- Result: people putting queries in POSTs
  - application-specific and frameworks
  - frameworks doing this leads to gratuitous tunnelling
  - HTTPbis recommendation likely to be around 8k

# Headers

- Some length limits (e.g. 20k total in Squid)
- Almost no-one handles line continuations
  - Result: effectively profiled out
  - Disallowed by latest HTTPbis changes
- Connection header control: not great
  - Result: extending protocol difficult
- Trailers aren't well-supported at all
  - Result: debug, status more difficult

# Partial Content

- Content-Range / 206
- Biggest use: PDF
- Some caches don't store partial content
  - e.g., Squid
- Flash URL API can access ranges, but VideoPlayer, etc. don't use it
- Result:

```
$vidID = $_GET["vidID"];
$vidPosition = $_GET["vidPosition"];
```

# Redirection

- Most* current browsers will redirect POST when they get a 307 Temporary Redirect
  - ... but not PUT or DELETE
  - ... and not a 301 or 302
  - * except Safari - it doesn't even do 307
- This is relatively new
- Result: login and lose your POST body

# Caching

- Basic conformance is there
  - max-age, no-cache, no-store, Expires, IMS, INM
- Invalidation isn't implemented*
  - Result: don't see your blog comments
- Updating headers on 304 and HEAD is spotty
- Warnings aren't generated
- Curl sends `Pragma: no-cache` by default
- Result: Opportunity cost

# Connection Handling

- Browsers limited to two concurrent connections to each server
  - ouch!
  - Result: BATCH, hosting on multiple names, etc.
- Being fixed in HTTPbis

# Pipelining

- Clients
  - Only Opera does by default (lots of heuristics)
  - The brave can turn it on in Mozilla
  - A few libraries allow (e.g., Serf)
- Most intermediaries will be OK with it, but won't forward
- Many servers handle it just fine; a few don't
- Risks: interleaved or out-of-order responses
- Predominant use today: SVN (thanks to Serf)
- Result: "waterfall" of requests; CSS spriting

# The Cookie Cesspit

- There is no cookie specification.
  - Netscape isn't complete
  - RFC2109 doesn't reflect current practice
    - Opera only major implementation of RFC2965
- Parsing raw dates is painful
  - `Set-Cookie: a=1; Expires=Thu, 24 July 2008 00:00:00`
  - requires special case handling
- Result: libraries required.

# Where Next for HTTP?

# Tests, tests, tests

- Most knowledge today is ad hoc
  - Some tools (e.g., co-advisor)
- Needed:
  - open source test framework
  - common test corpus
- messaging, semantics...
- For clients, intermediaries, servers and caches

# Authentication

- Basic is interoperable, but not secure
- Digest is more secure, but not terribly interoperable
- Many newer requirements not addressed
  - Phishing
  - Delegated auth
- OAuth BoF last week in IETF Minneapolis
- Other efforts still coalescing

# Better Transport

- head-of-line blocking STILL an issue
  - Pipelining isn't well-supported, and doesn't completely solve the problem
- HTTP doesn't guarantee integrity
  - except with Content-MD5 (which no one does)
- HTTP-over-SCTP
  - Great for lossy / long-distance networks
  - proxy-to-proxy overlays
  - uDel, Cisco

# PATCH

- "Restful" APIs are starting to abuse PUT
  - "update that with this..."
- PATCH allows you to apply a diff to a resource
- Currently an Internet-Draft

# Prefer Header

- Lets a client state what it wants;
  - Full content in response body
  - Status message in response body
  - No response body
- E.g., POST /order-handler
- Currently a (quiet) Internet-Draft

# Link Header

- Under-developed part of the Web arch: typed links
- Advertise/discover links in HTTP headers
  - “this invalidates <foo>”
  - “the previous one is <bar>”
  - “edit this over at <baz>”
- In RFC2068, taken out of RFC2616
- Bringing back as an internet-draft

# Caching Refinements

- stale-while-revalidate
  - hide server latency from clients
- stale-if-error
  - hide server errors from clients
- Out-of-band change monitoring
- Using resource relationships to invalidate
- Explicit cache key

# HTTP Software

- Higher-level (but still RESTful) abstractions
  - e.g., webmachine
- Better feature set coverage
  - e.g,. Rack::Cache
- Intermediary building blocks
  - high performance/concurrency
  - e.g., xLightweb

# A Word on O2.0

- OpenID, OAuth, XmlHttpRequest, HTML5, Comet, "reliable" HTTP, BATCHing, Gears...
- General tendency to
  - use least-common-denominator tech
  - use familiar tools
  - ignore intermediaries
  - fail to consider overall architecture
- High opportunity cost

# Take-Aways

- Implementations are (obviously) usable, but
  - They sometimes impose arbitrary limits
  - They don't expose some important controls
- Developers will always take the easiest path
- Not implementing because no-one uses it is a self-fulfilling prophesy
- HTTPbis is an opportunity to
  - get implementers together
  - clarify ambiguities
  - improve interop
  - make HTTP a more stable basis for the next 10+ years

# Resources

- http://tools.ietf.org/wg/httpbis/
- http://tinyurl.com/65e9lb [ implementation sheets ]
- http://coad.measurement-factory.com/
- http://www.mnot.net/blog/2007/06/20/proxy_caching
- http://www.mnot.net/blog/2006/05/11/browser_caching