The future ain't what it used to be

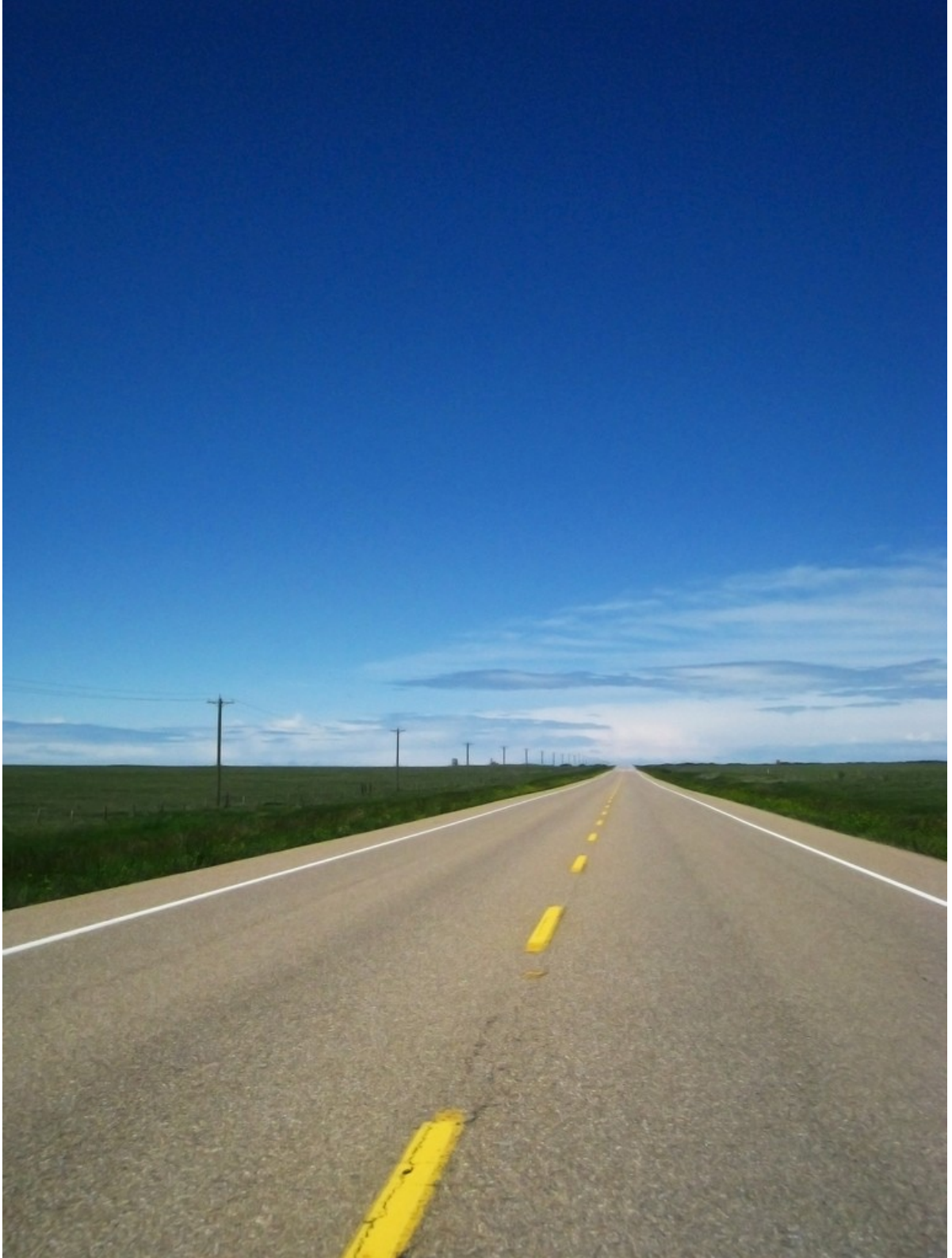
Poul-Henning Kamp

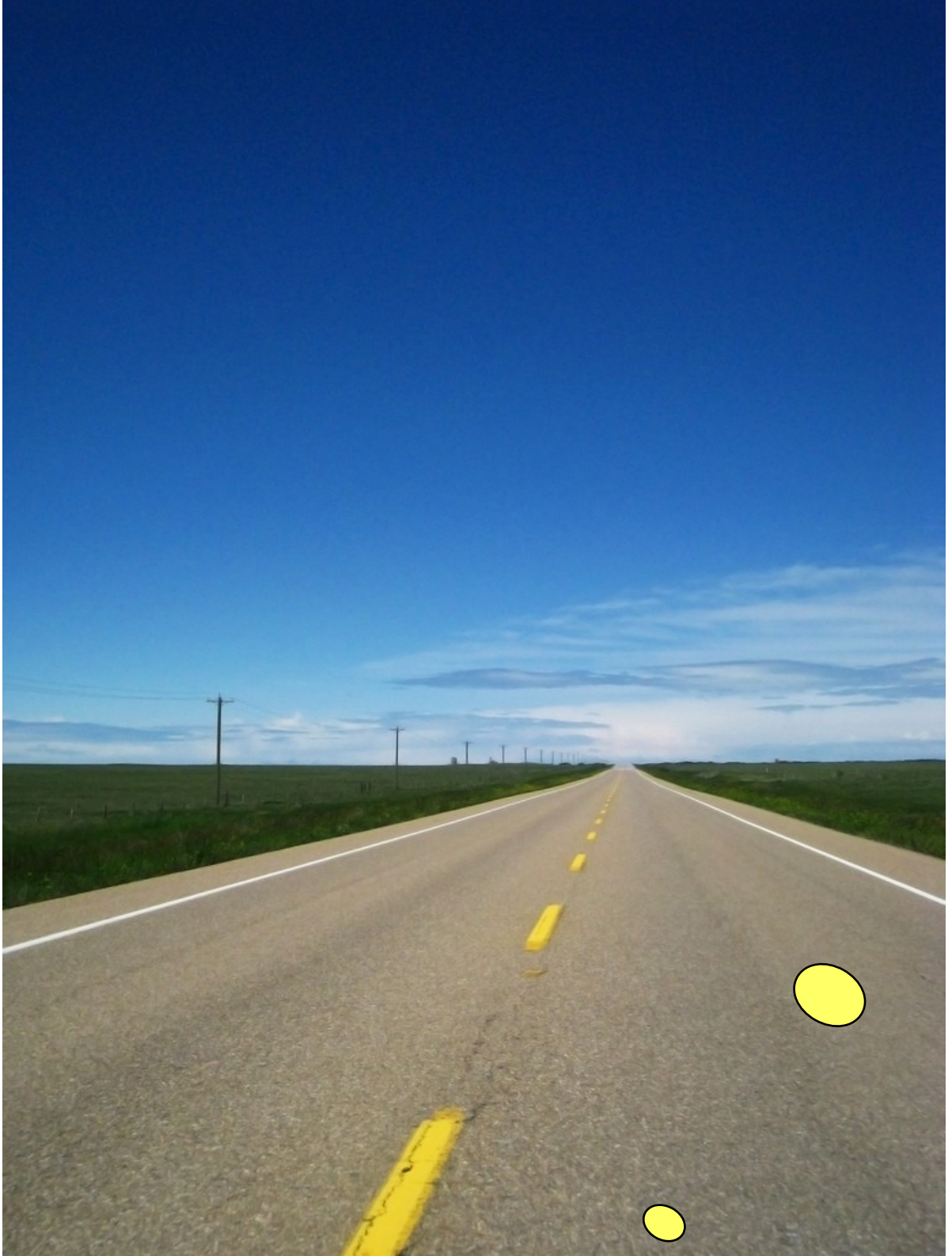
phk@FreeBSD.org

phk@Varnish.org

@bsdphk

Contents



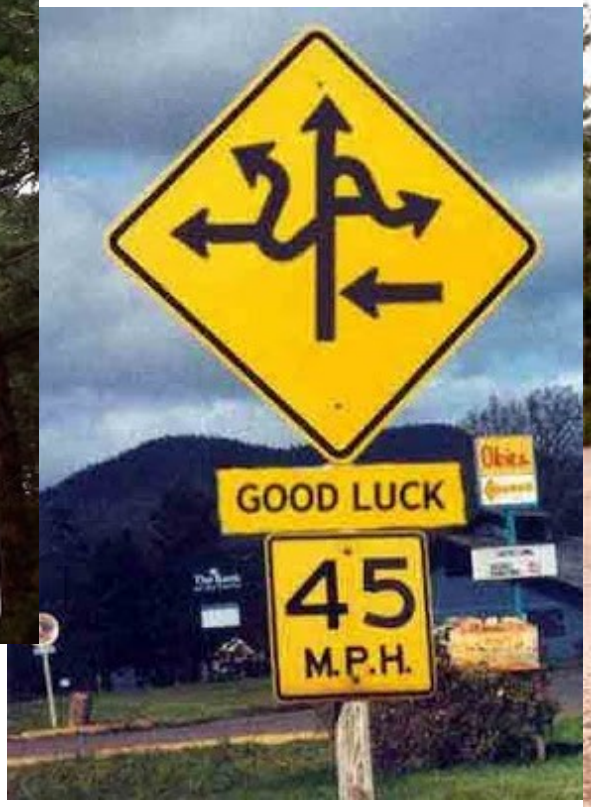

Booooooring…
I bet even a computer could do this…


GM
CAT
19
CATERPILLAR
Carnegie Mellon
Continental
NetApp
Google
APPLANIX
vector
ibeo
MOBILEYE
intel


Brentwood
Chipping Ongar
Industrial Estates
A 128
Secret Nuclear Bunker


Table Mountain National Park
Warning
Please look under your vehicles for penguins
Sponsored by Boulders Beach Lodge


RIGHT LANE
MUST
RIGHT LEFT


GOOD LUCK
45
M.P.H.

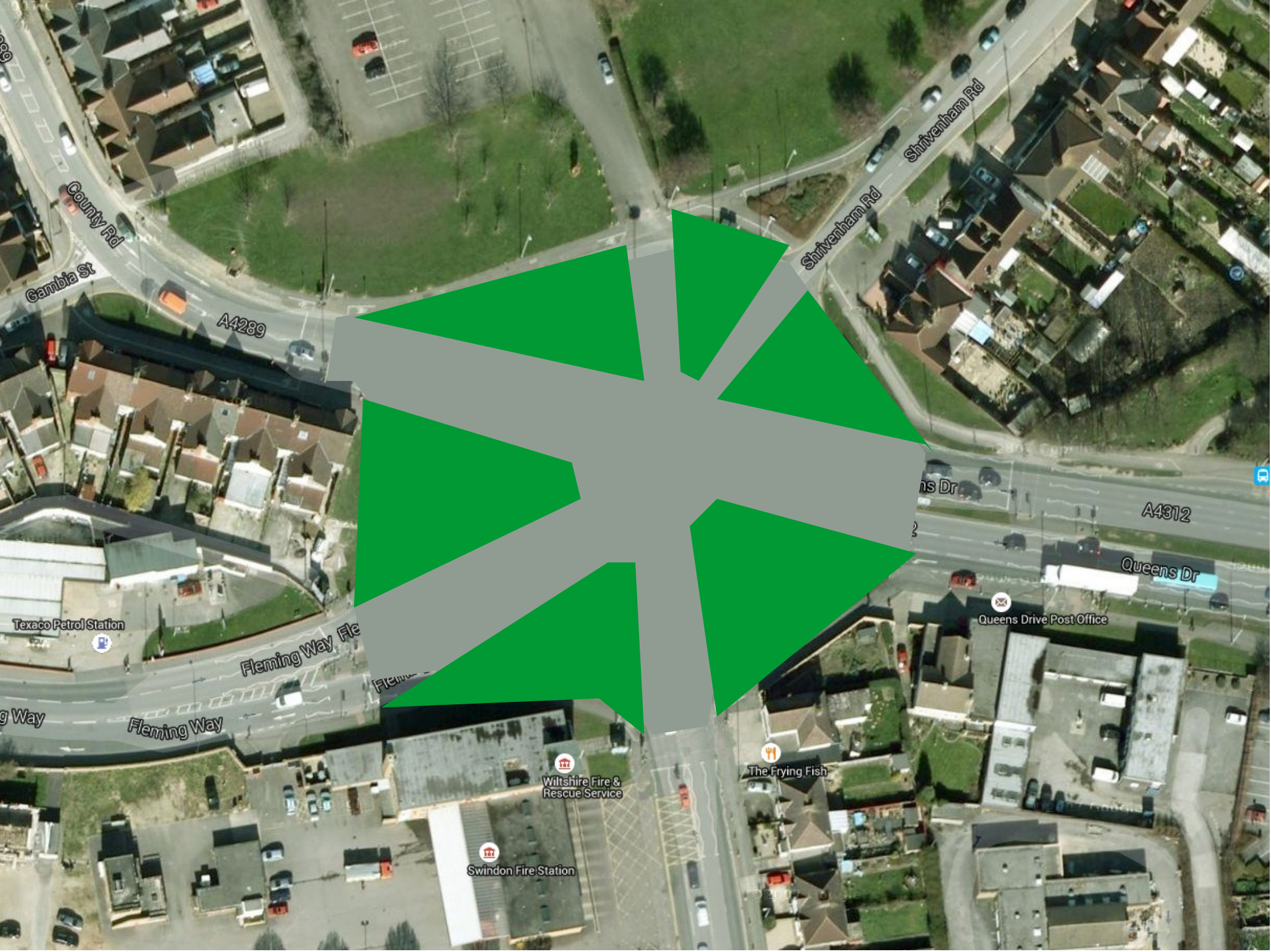

STOP
WRONG WAY
DO NOT ENTER


EXIT 90
REST ROOMS
7:00 AM - 11:00 AM ET
EXIT 3/4
OLD 49
4 EAST
EXIT 74A
4
Kissimmee
2 MILES
EXIT 62
195
EXIT 351B
TO 10 WEST
1 3/4 MILES
EXIT 62
2 PERSON CAR POOLS ONLY
H H EXIT 84
Ivanhoe Blvd
1 MILE
EXITS 64AB
EXIT 62
EXIT 62
OLD 24CDE
EXIT 62
TOLL 536
441
46A
TOLL 95
TOLL 417
NORTH EAST
Sand Lake Rd.
Dora
John
Wood
Fairbanks
Hell's
Celebration
EXIT ONLY
HOV
EXIT ONLY
OLD 24CDE
RIGHT LANE ENDS
1/4 MILE
Eatonville
Historic Dist.
20
REST AREA
TO
S. International Dr.
Orlando World Center
EXIT 1031 3/4


THE MAGIC ROUNDABOUT
Ring road
Cirencester
A4289
(M4)
Town
centre
Marlborough
Burford
Oxford
A&E
A4312
Gorse Hill
North Star


County Rd
Gambia St
A4289
A4289
A4289
The Magic Roundabout
Shrivenham Rd
Shrivenham Rd
Queens Dr
A4312
A4312
Queens Dr
Queens Drive Post Office
Texaco Petrol Station
Fleming Way
Fleming Way
Fleming Way
Fleming Way
Fleming Way
B4289
Wiltshire Fire & Rescue Service
Swindon Fire Station
The Frying Fish


Volkswagen Automotive
Innovation Laboratory


County Rd
Gambia St
A4289
Shrivenham Rd
Shrivenham Rd
A4312
Queens Dr
Queens Drive Post Office
Texaco Petrol Station
Fleming Way
Fleming Way
Wiltshire Fire &
Rescue Service
Swindon Fire Station
The Frying Fish

HTTP — Back to the future…

Fundamentally a RPC request:

```
client → server   ”Please ?”
server → client   ”My pleasure!”
```

Used to be:

Dumb client & Smart server

Now:

Smart client & ______ server ?

The protocol moves

HTTP used to transfer:

Images, HTML, CSS, JS

- predigested/completed data
- JS+CSS improved presentation

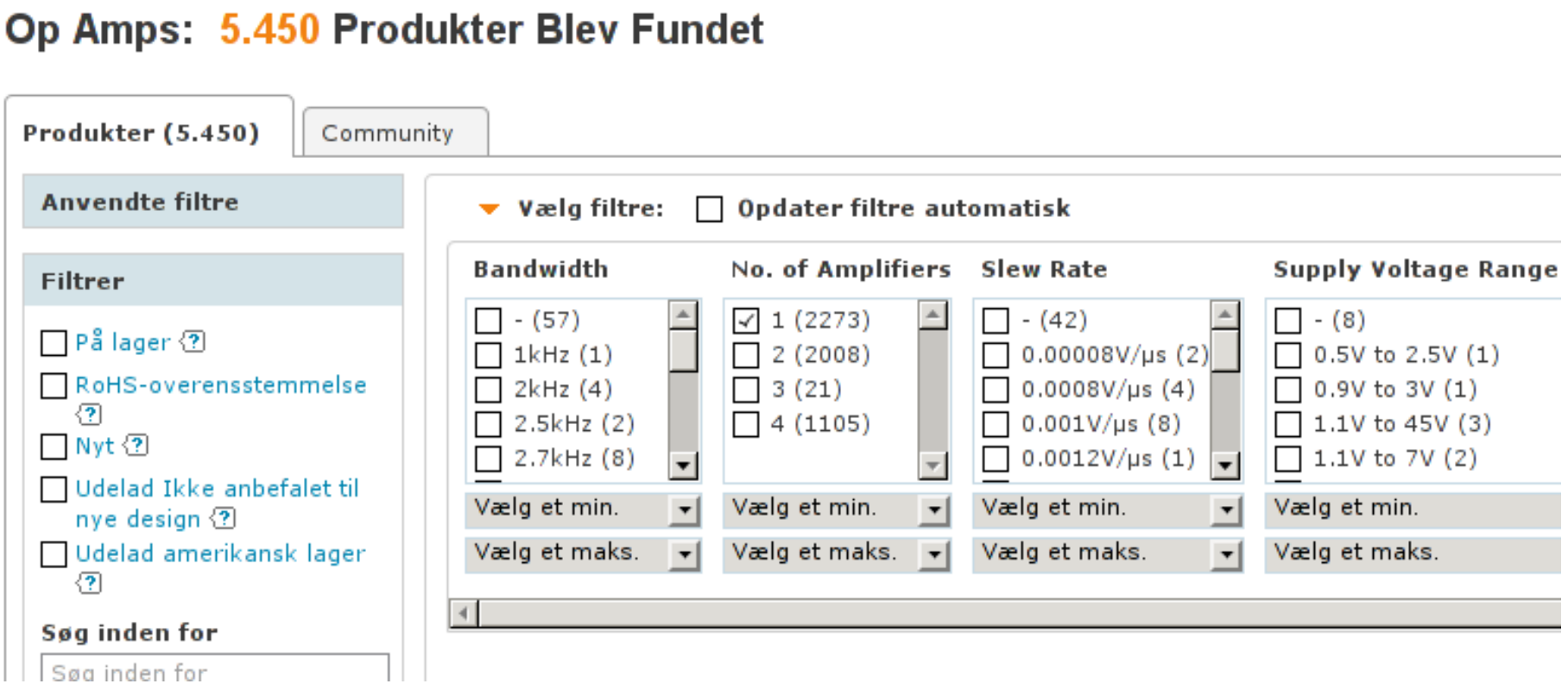
Now:

Images, JS, Data, HTML, CSS

- raw data
- JS processes data on client
- HTML+CSS is left as scaffolding

Consequences

Narrow view:
Better UX



Consequences

Narrow view:
- Better UX

Big view:

- Reduces total CPU load
- Reduces network traffic
- Shifts CPU load to device
- Shifts electricity bill to user

→ Moves cost of computing to decision maker

Pub-Quiz:

Bigger CO2-footprint?

1: Air traffic

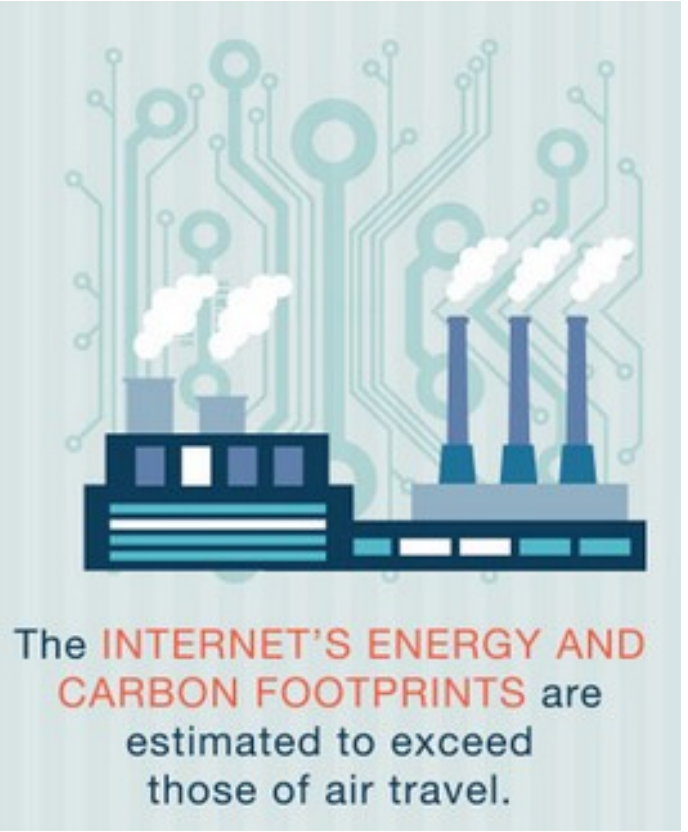
2: Telecommunication

Pub-Quiz:

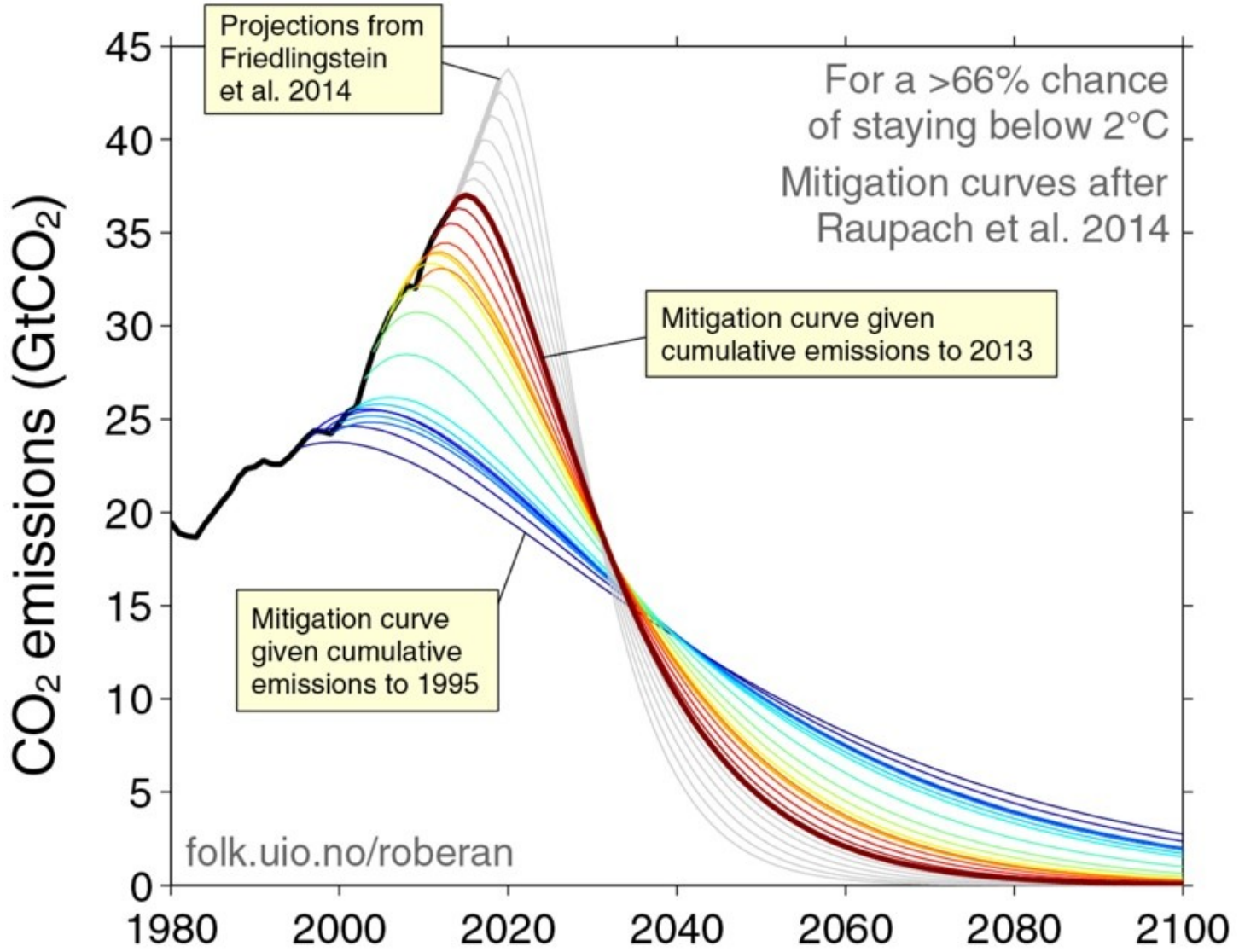
Bigger CO2-footprint?

1: Air traffic

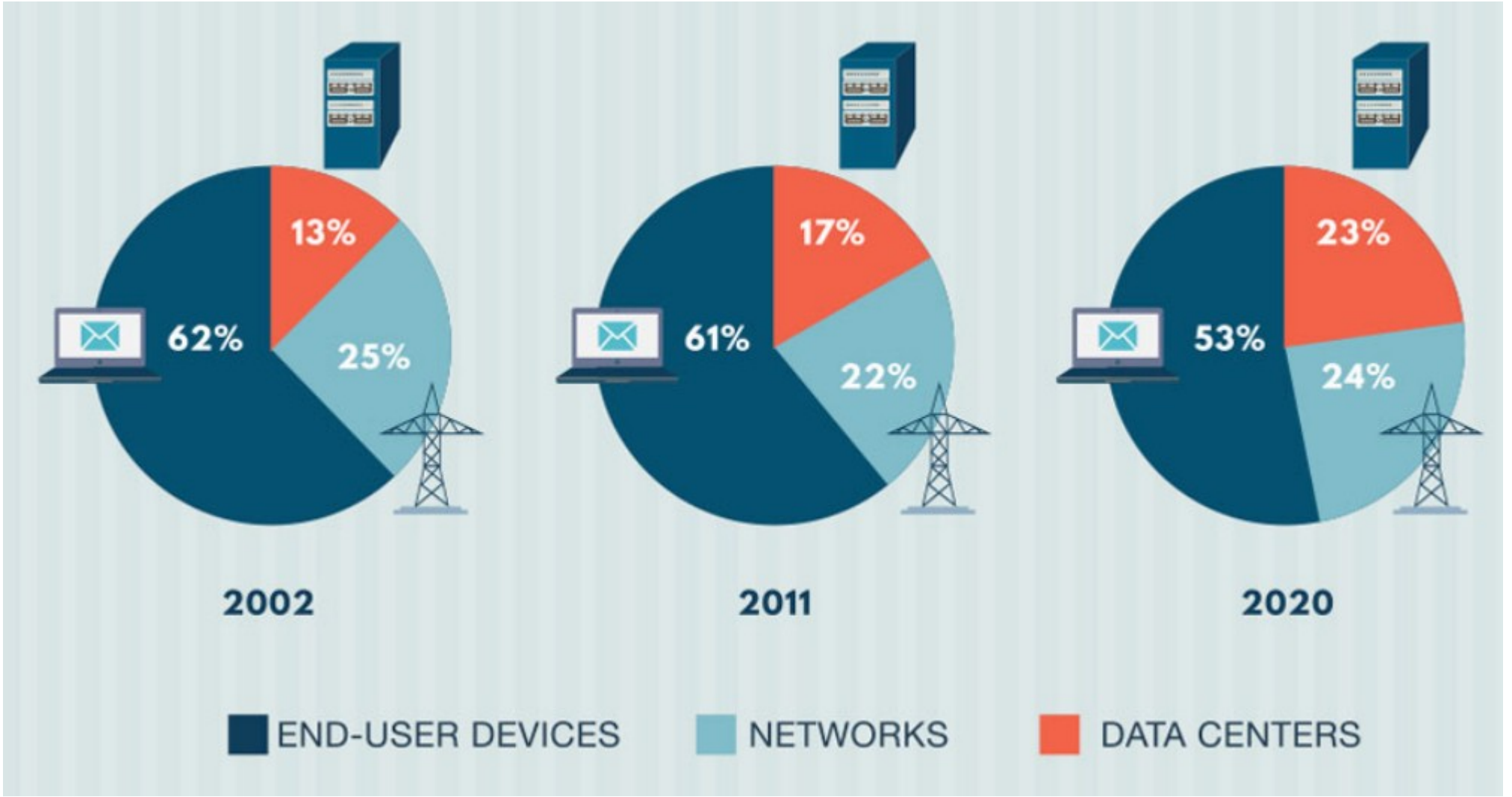
2: Telecommunication



PS: We only have 10 years…



Follow the electricity bill…



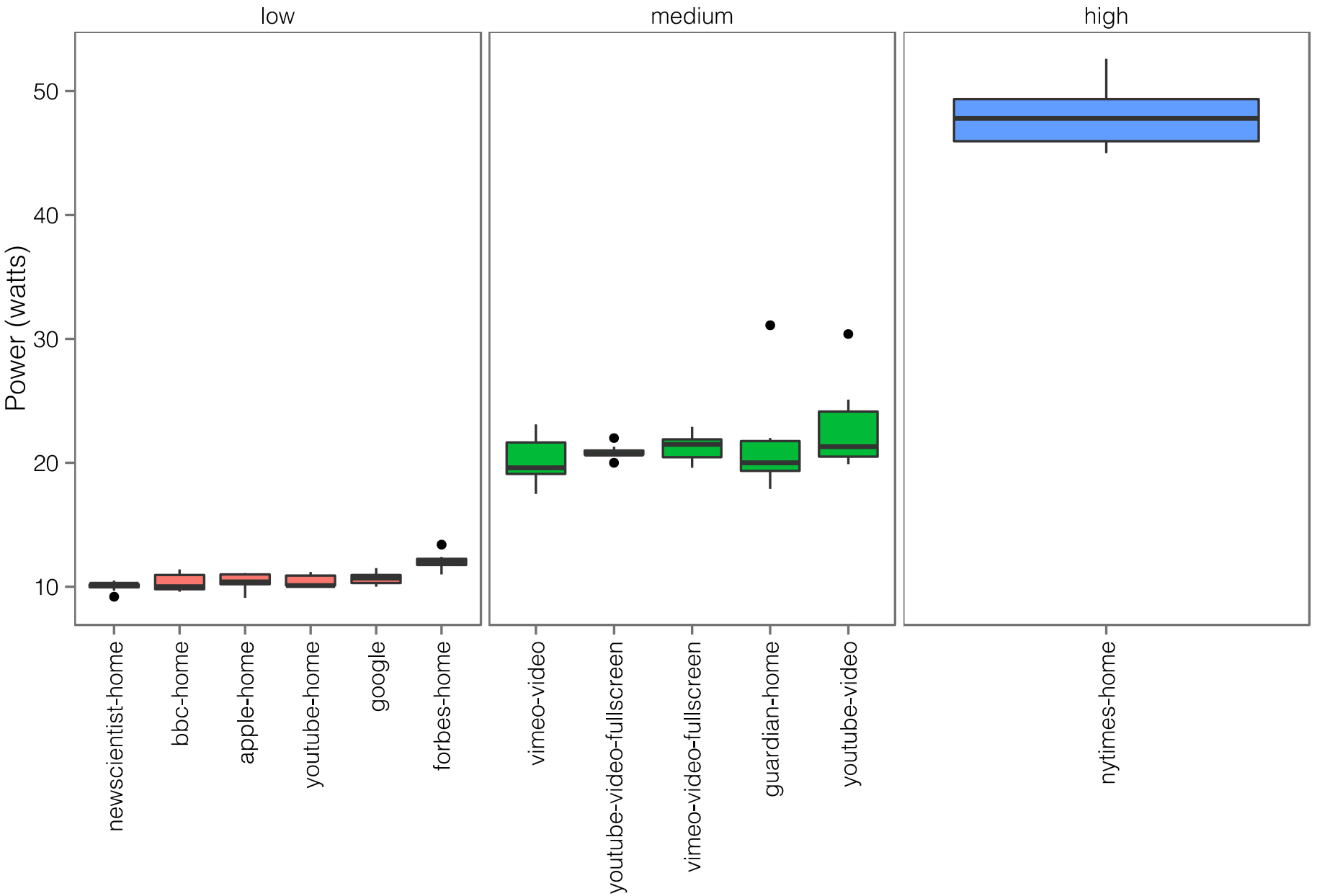
Source: http://climatecare.org/infographic-the-carbon-footprint-of-the-internet/

More consequences

End-user device runs on battery

Shitty computing will:

- Run battery down
- Make users unhappy
- Be complained about on social media



http://santtu.iki.fi/2015/06/18/browser-power-consumption/

Back of the envelope sanity-check

7:33 min avg time on site [alexa]

7.5 min * 40W = 5 Wh = 5g CO_2

30 M unique desktop/month

Guess: 10 visits per month avg.

30 M * 10 * 5g * 12 = 18,000 ton/year

~= 1 avg US person

~= 180 avg Malawi persons

HTTP, HTTP, HTTP, Eggs and HTTP

More mobile devices

More shitty connectivity

- Efficient transmission
- CDN's and edge-caching
  - → A cache on each mobile tower
- Integrity, authenticity
  - → Cacheable encrypted content

HTTP, HTTP, HTTP, Eggs and HTTP

Internet of Crap:

- Natural Gas-Meters run off batteries
  1. DO NOT EXPLODE!
  2. Do not require software updates
  3. Do not waste battery power
  4. Whatever

HTTP, HTTP, HTTP, Eggs and HTTP

API:

Move data between programs

A) Real-time
B) Bulk

→ Authentication, Integrity, Secrecy
→ Performance (RTT and BW)

HTTP, HTTP, HTTP, Eggs and HTTP

Browser:

- Render complex view as fast as possible
- → Fetch many resources in parallel
  - → Multiplexing, priority, flow-control
- Driving SPDY, HTTP/2, QUIC development
- Doesn't give a shit about non-browser HTTP

So what exactly is HTTP anyway ?

Sort of a mess… (But you knew that)

HTTP was a semantic protocol

Became a transport protocol with semantics

Became **THE** transport protocol
   (WebSocket etc: ”Nevermind the semantics”)

And then it became two transport protocols

… with almost identical semantics

Now also a different transport under the sematics

TCP is just wrong for browsers

TCP was for bulk transfer of huge objects
   FTP, email, usenet, HTTP/0

Added workarounds for interactive traffic:
   TELNET, X11

Very slow development cycle

   Break TCP and **nothing** works…

What browsers want

Today browsers need many objects

→ Multiplexing

Per stream prioritization & flow-control

HTTP/1 hack: One TCP per stream

HTTP/2 mistake: All streams into one TCP

The QUIC fix: Drop TCP entirely

The revenge of WAP

In the future we will have ”kinds of HTTP”

HTTP/1.1
- Because there is so much crap out there already

Browser
- Because TCP really isn't what a browser needs

HTTP/3 ?
- Structure non-text headers ?
- Rigidly defined semantics
- Removing old cruft
- Add ”session” concept
- Remove security mistakes (aka: Cookies)

Divorcing Semantics and Transport

Transport:

- Move {Envelope (=top_line), Headers, Object}

Semantics:

- What does Envelope and Headers mean
  and what do they say about Object

Should have been done before HTTP/2

- but HTTPbis/RFC723x messed that up

The HTTP Political CABAL

Significant perverse incentives

Browsers have defacto veto power

- Browsers funded by privacy invasion

Google can do whatever they want

- They have their own browser

Impossible to prove NSA-noninvolvement

SSL/TLS

Still the wrong solution

- But the solution we have right now

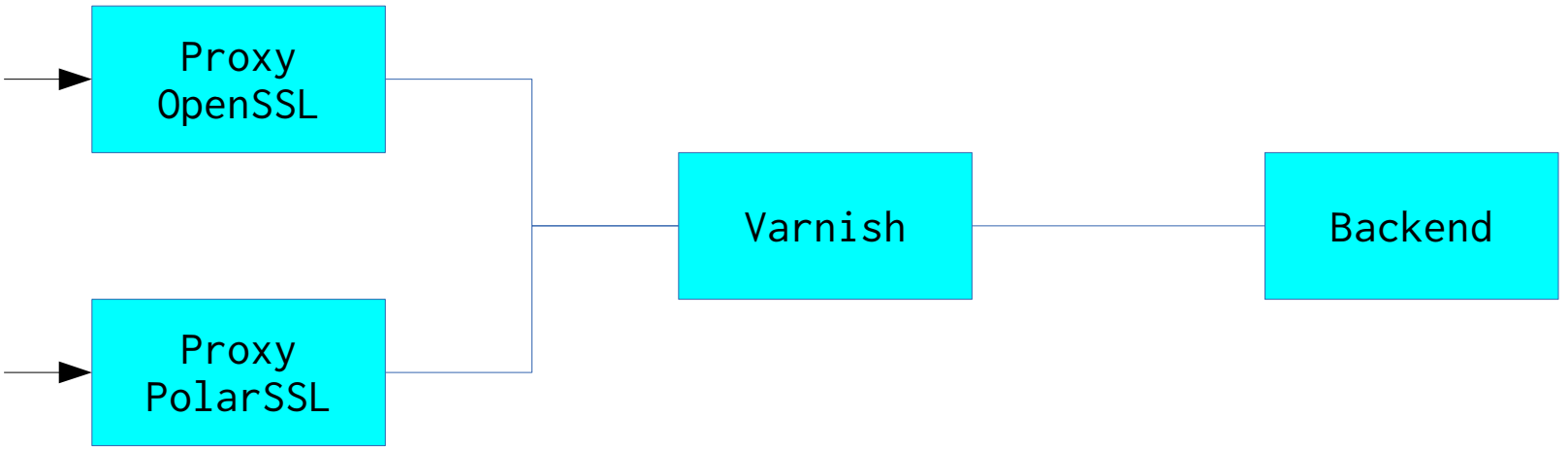
Best solution: draft-thomson-http-encryption

- Encrypt inside the envelope
- No 3-way handshake needed
- Lawful traffic logging possible
- Lawful filtering possible (based on envelope)
- Caching possible

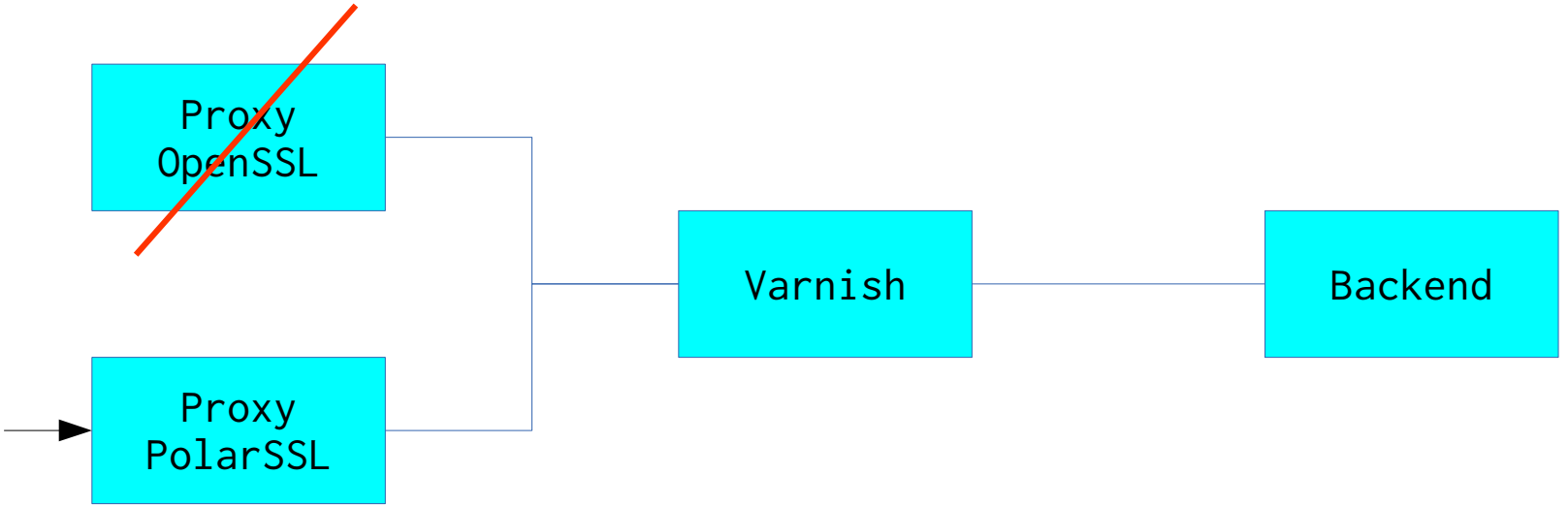
SSL/TLS in Varnish

Still a bad idea IMO

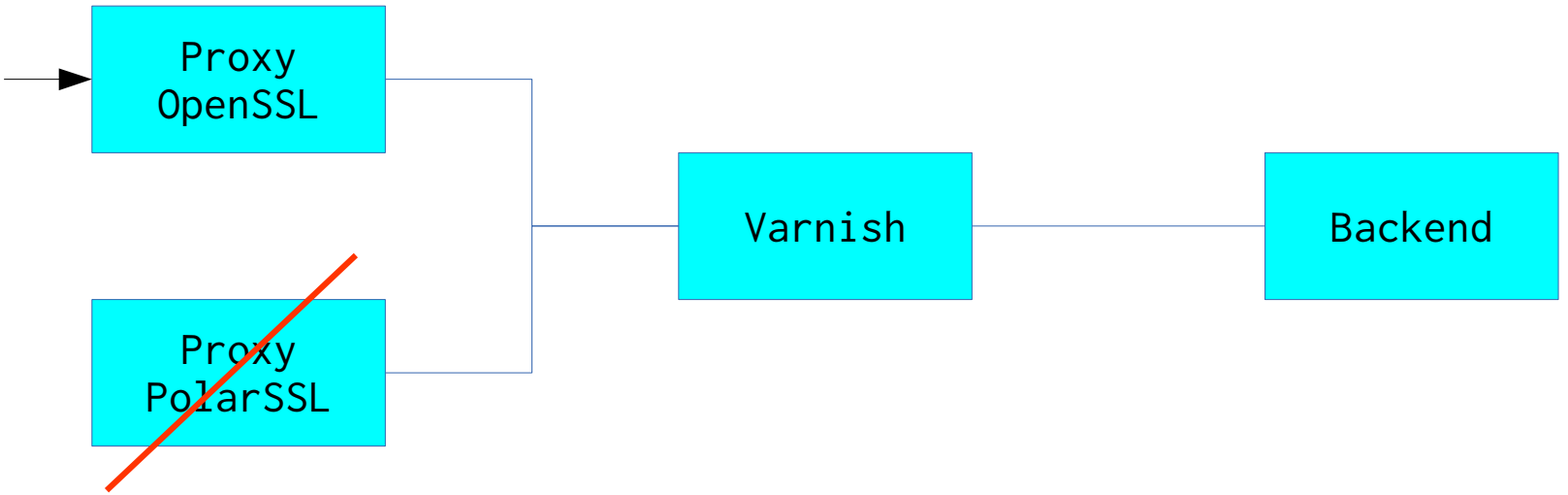
Do it this way:



Even weeks: OpenSSL broken



Odd weeks: PolarSSL broken



Varnish 4.1

Improved jailing of worker process
  ”varnish” and ”vcache” uids

Better workspace overflow
  500 rather than panic

PROXY protocol support

Varm/Cold VCL states

VMOD defined backends

Tons of minor stuff & bug-fixes

Varnish: Quality, Speed, ~~On Time~~

Quality:

9¾ Years with no major security panics

Speed:

You know that

On Time:

We have never released on the planned date

Varnish 4.1

Why did it take so long ?

- DevOps promoted to Developers
- Getting live tests became MUCH harder
- Release waited 4-6 months for live tests

Not happening again…

Future ”Head” Varnish releases (a.b.0)

We will do two releases per year

- Spring & Autumn, (dates TBD)
- Whatever is ready is in the release
- Whatever is broken is in the release

”Real-life-test” now devops/users responsibility

- The developers have nowhere to do that

Future maintenance releases (a.b.X)

Not all ”head” releases will be ”blessed”

- Case by case decision

Blessed releases will have follow up releases

- As warranted and as time permits

Varnish project leaves home

Varnish Software has hosted project until now
   (Cue: Spontaneous applause)

They're growing and we're cramping their style
   Almost only server left on premise

Also some commercial leakage → project homepage

Varnish project moves

Trac → GITHUB

- In progress

New project homepage

- Owned by devs & community

New project server

- In progress

Varnish Usage Statistics

Download Lead List

Websites using Varnish

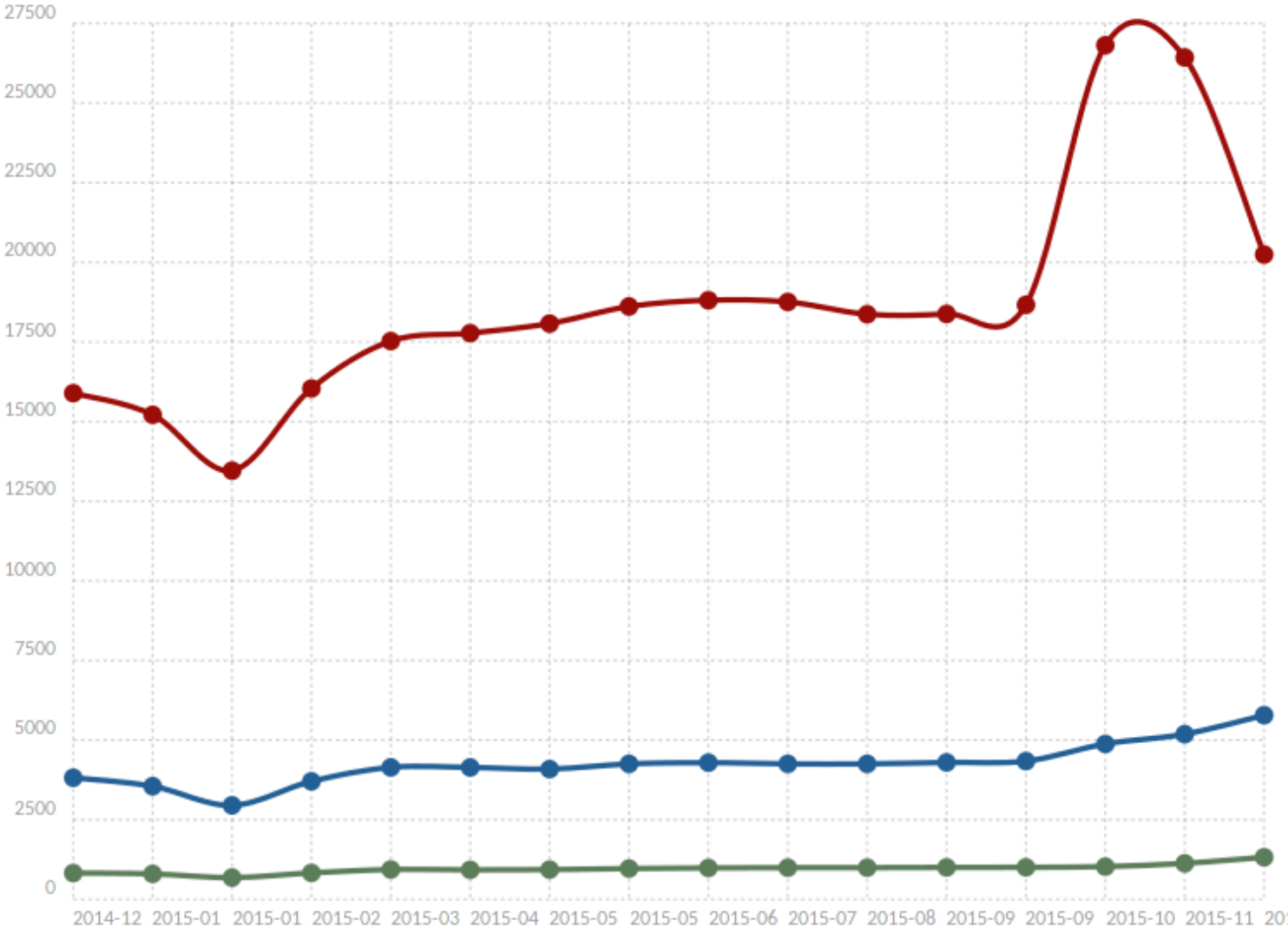


Chart Data

Source	Legend	Chart
Top 10k		☑
Top 100k		☑
Top 1m		☑
Internet		☐

Coverage Totals

Quantcast Top 10k	1,322 of 10,000	13.2%
Quantcast Top 100k	5,781 of 100,000	5.8%
BuiltWith Top Million	20,243 of 950,848	2.1%
Entire Internet	2,233,919 of 348,410,573	0.6%

Varnish Moral License

Varnish cache development brought to you by:

- Fastly
- Varnish-Software
- UPLEX
- ”ADJS”

Much appreciated!