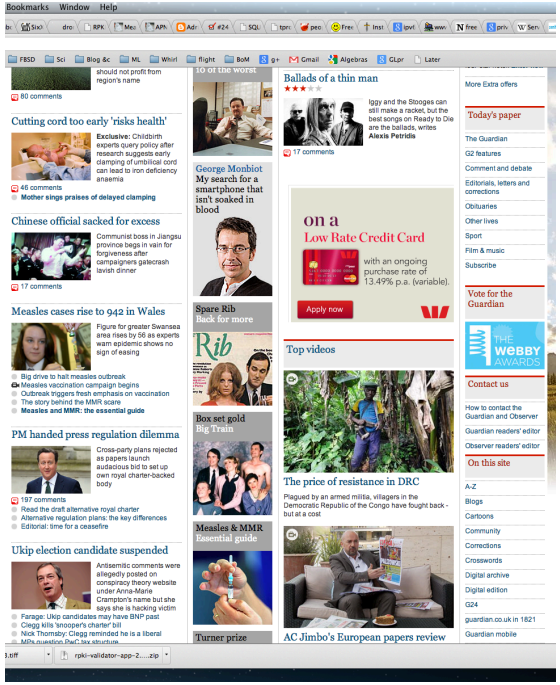
Measuring DNSSEC on the client

George Michaelson
Geoff Huston

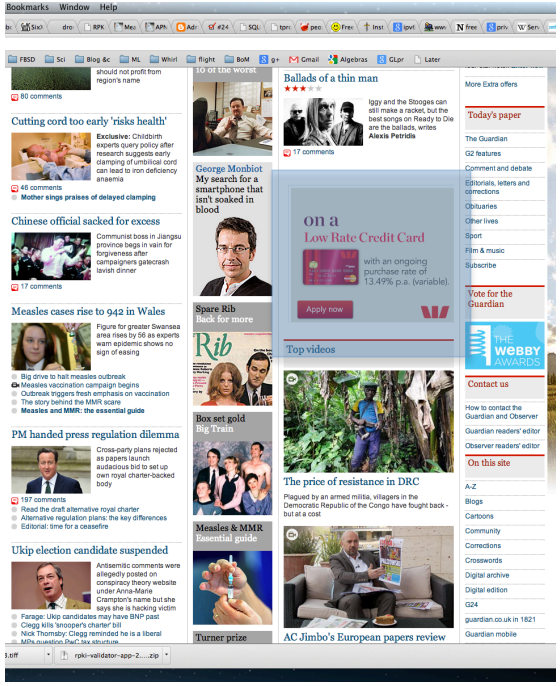
Overview

- ~5 million measurements
  - Internet-wide, 24/7
- Browser-embedded flash
  - Good DNSSEC signed fetch
  - Broken DNSSEC signed fetch
  - Unsigned fetch
- What do clients see?
  - Who do clients use to do resolving?
- How are we going with DNSSEC deployment?

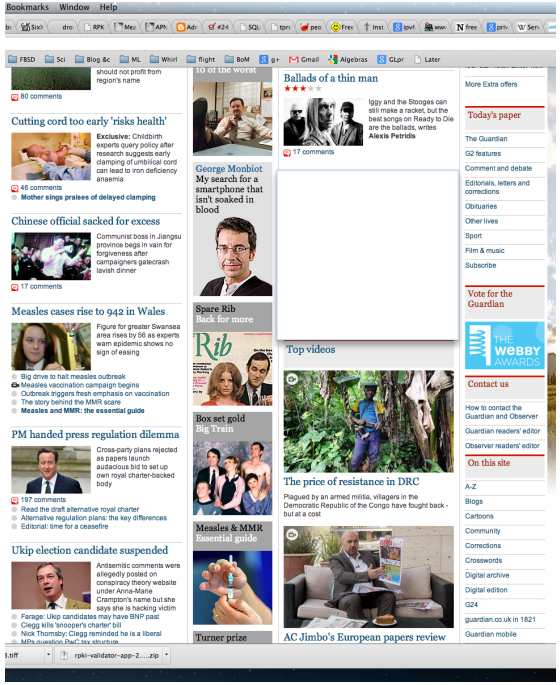
Flash inducted web clients



Flash inducted web clients

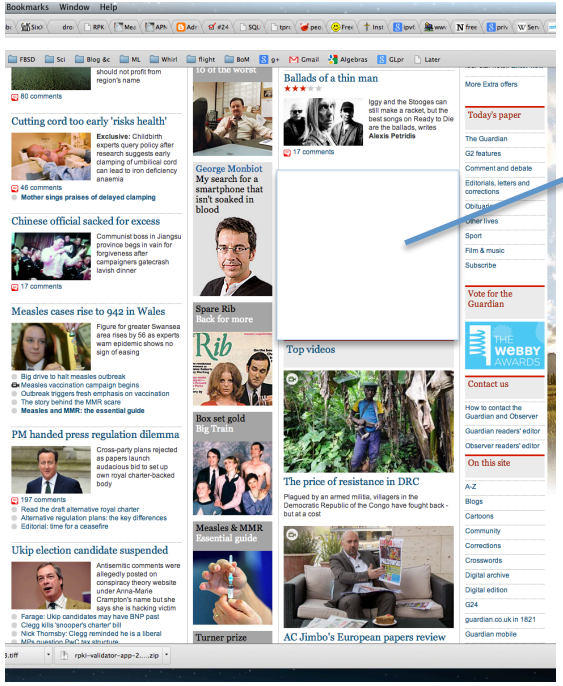


Flash inducted web clients

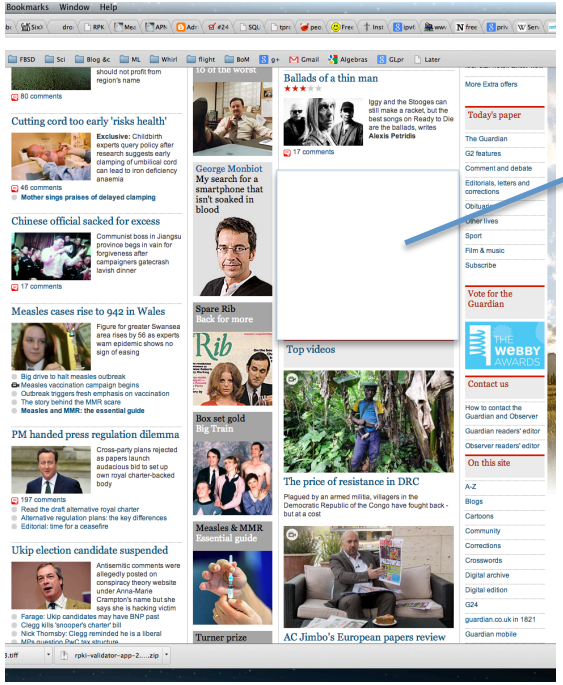


"hmm I could sell this space"

Flash inducted web clients



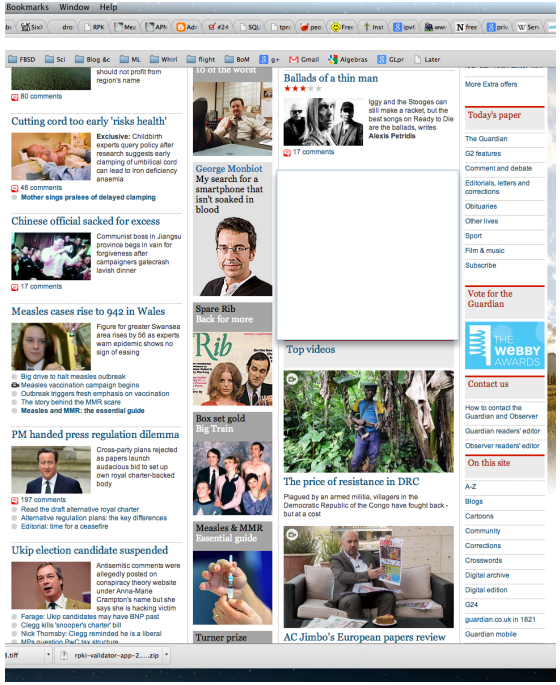
Flash inducted web clients

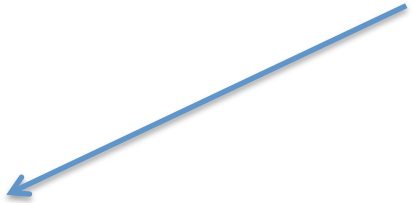


"ad space for sale"

Ad Feed

Flash inducted web clients

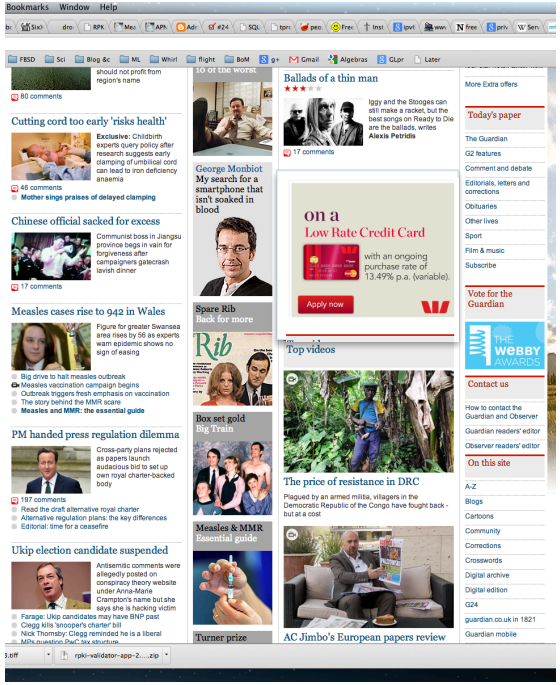




Ad Feed

"here's an ad
And 50c to show it"

Flash inducted web clients



Ad Feed

"I showed your ad: give me a dollar"

Ad Source

flash

- Ads written in flash to get interaction
  - Hover-overs animate
  - Visual, audio assets loaded off the net
- Flash is actionscript, full programming language
  - Threaded
  - Supports network calls, gethostbyname()

APNIC's measurement technique

- Craft flash/actionscript which fetches network assets to measure.
- Assets are reduced to a notional '1x1' image which is not added to the DOM and is not displayed
- Assets can be named (gethostbyname()) or use literals (bypass DNS based constraints)
- Encode data in the name of fetched assets
  - Result is returned by DNS name with wildcard

APNIC’s ad

APNIC’s ad

APNIC's ad

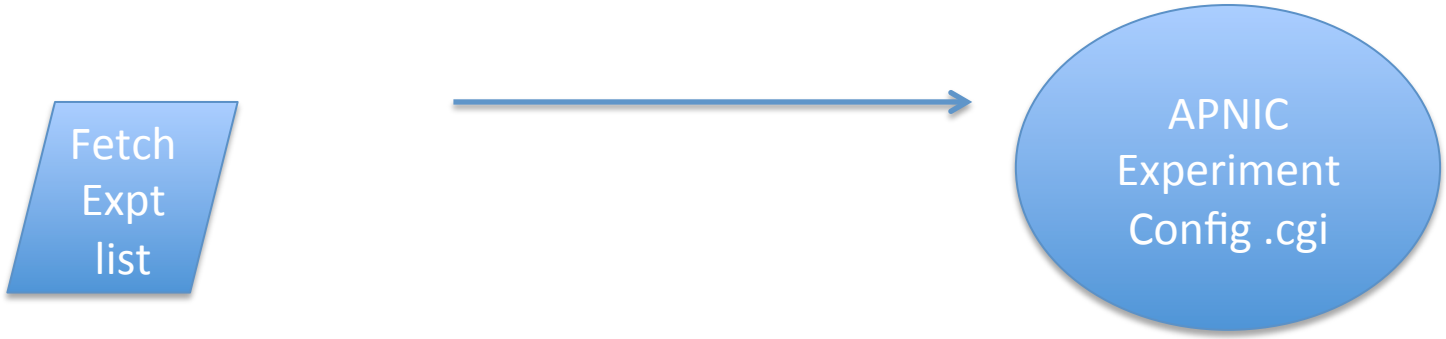
Standard 480x60 size, fits banner slot in most websites
Deliberately boring, to de-preference clicks (cost more)

APNIC's ad

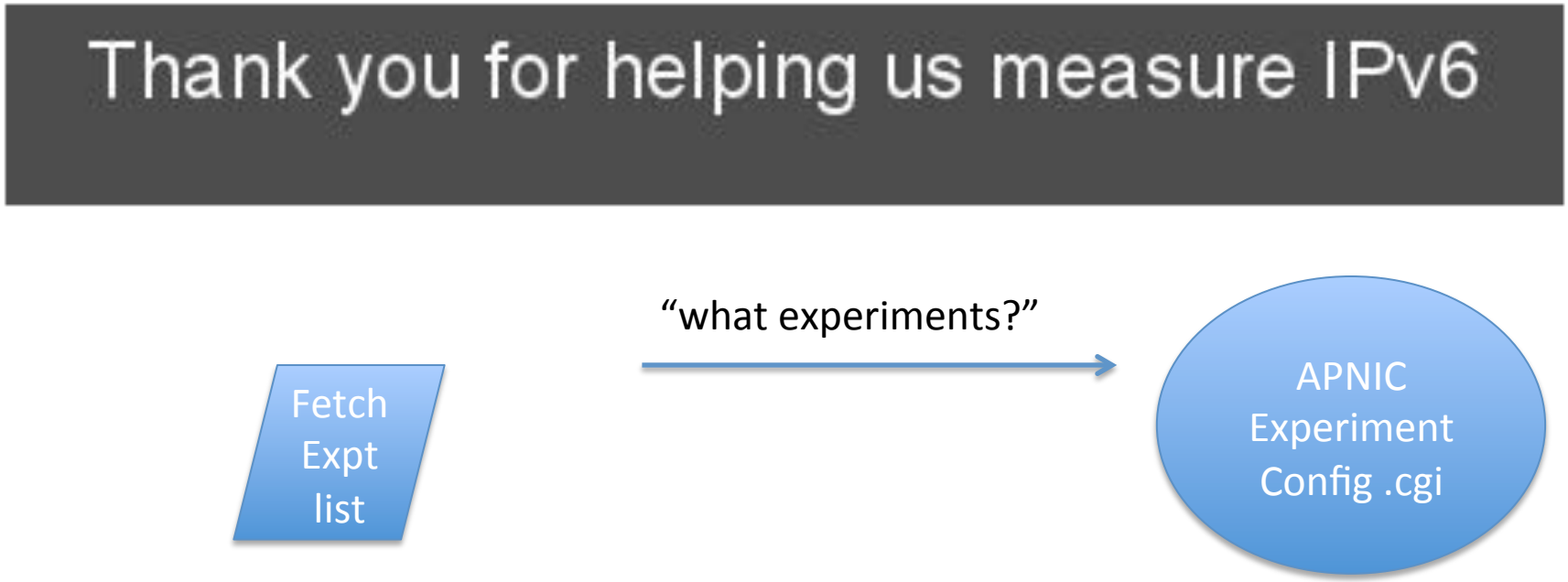
APNIC's ad

APNIC's ad

Thank you for helping us measure IPv6



APNIC's ad

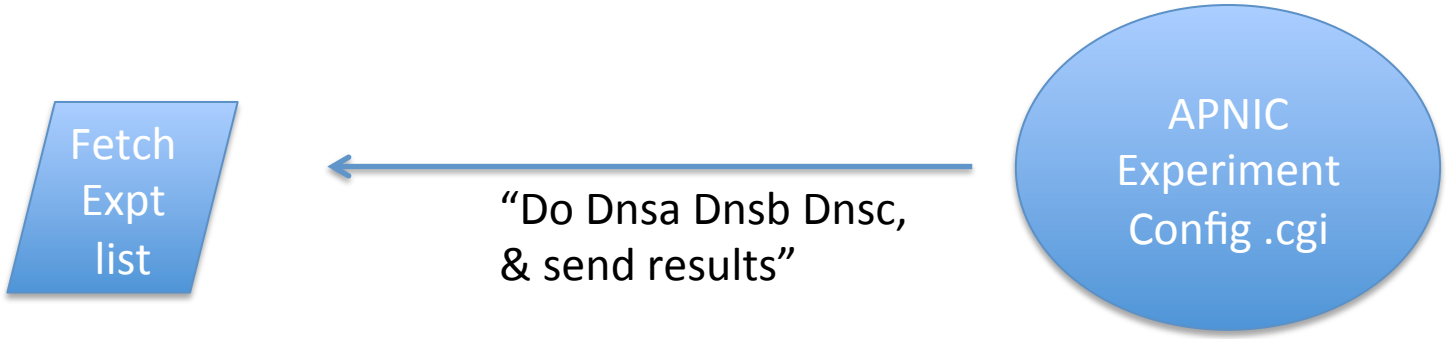

Thank you for helping us measure IPv6
"what experiments?"
Fetch Expt list
APNIC Experiment Config .cgi

APNIC's ad

Thank you for helping us measure IPv6

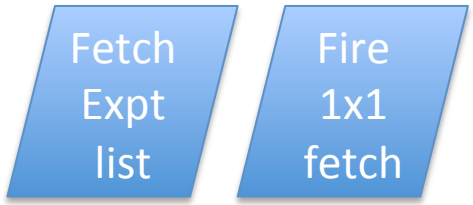
APNIC's ad

Thank you for helping us measure IPv6



APNIC's ad


Thank you for helping us measure IPv6


Fetch
Expt
list
Fire
1x1
fetch

APNIC's ad

Thank you for helping us measure IPv6

APNIC's ad

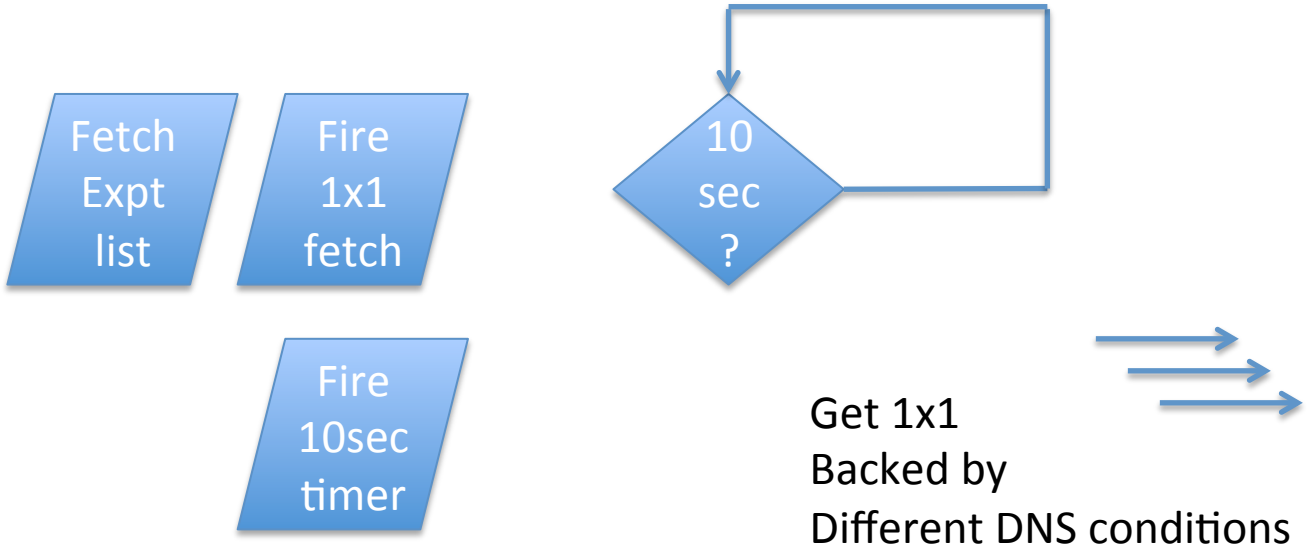
Thank you for helping us measure IPv6

APNIC's ad


Thank you for helping us measure IPv6
Fetch Expt list
Fire 1x1 fetch
Fire 10sec timer

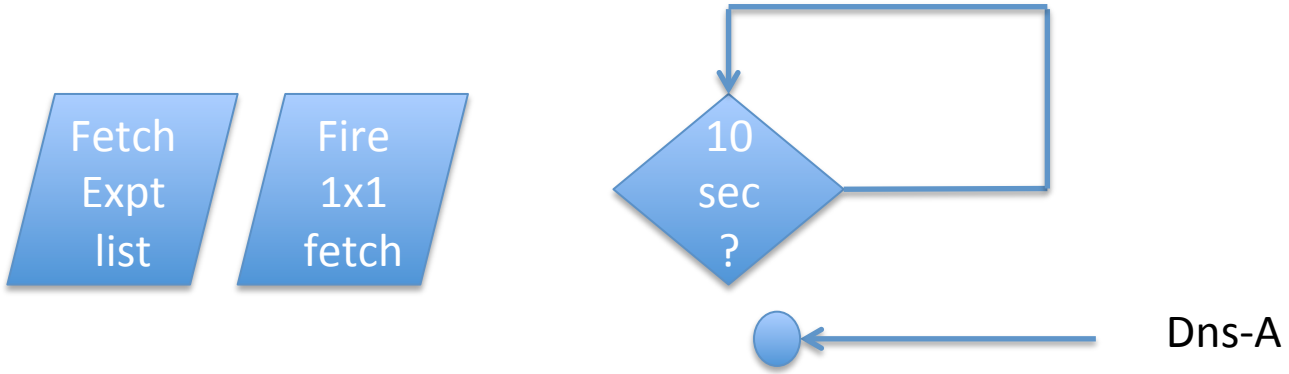
APNIC's ad

Thank you for helping us measure IPv6



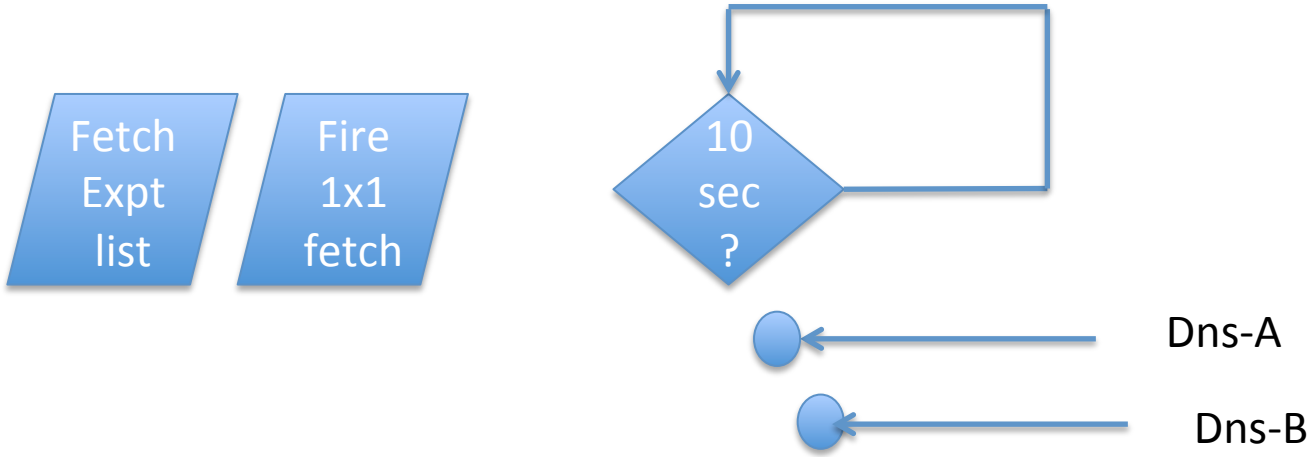
APNIC's ad

Thank you for helping us measure IPv6



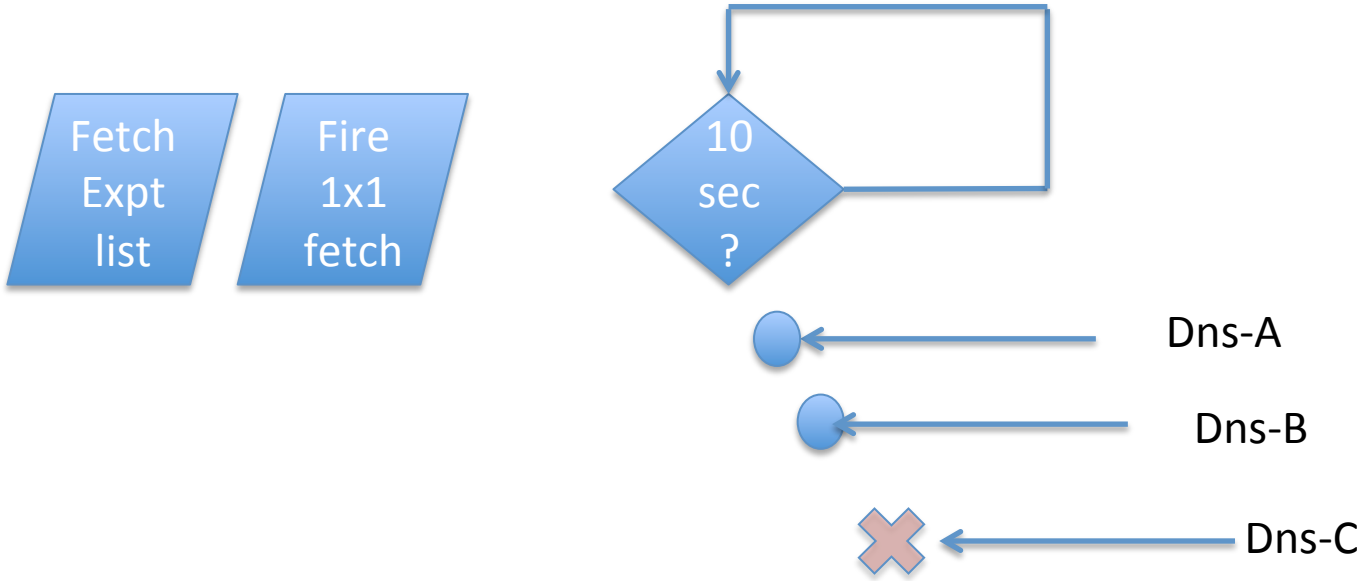
APNIC's ad

Thank you for helping us measure IPv6



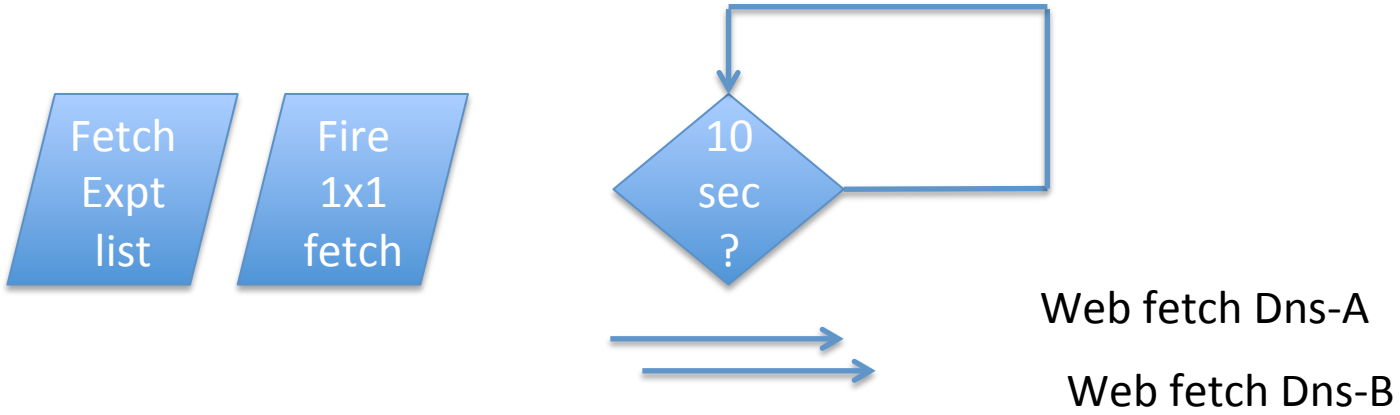
APNIC's ad

Thank you for helping us measure IPv6



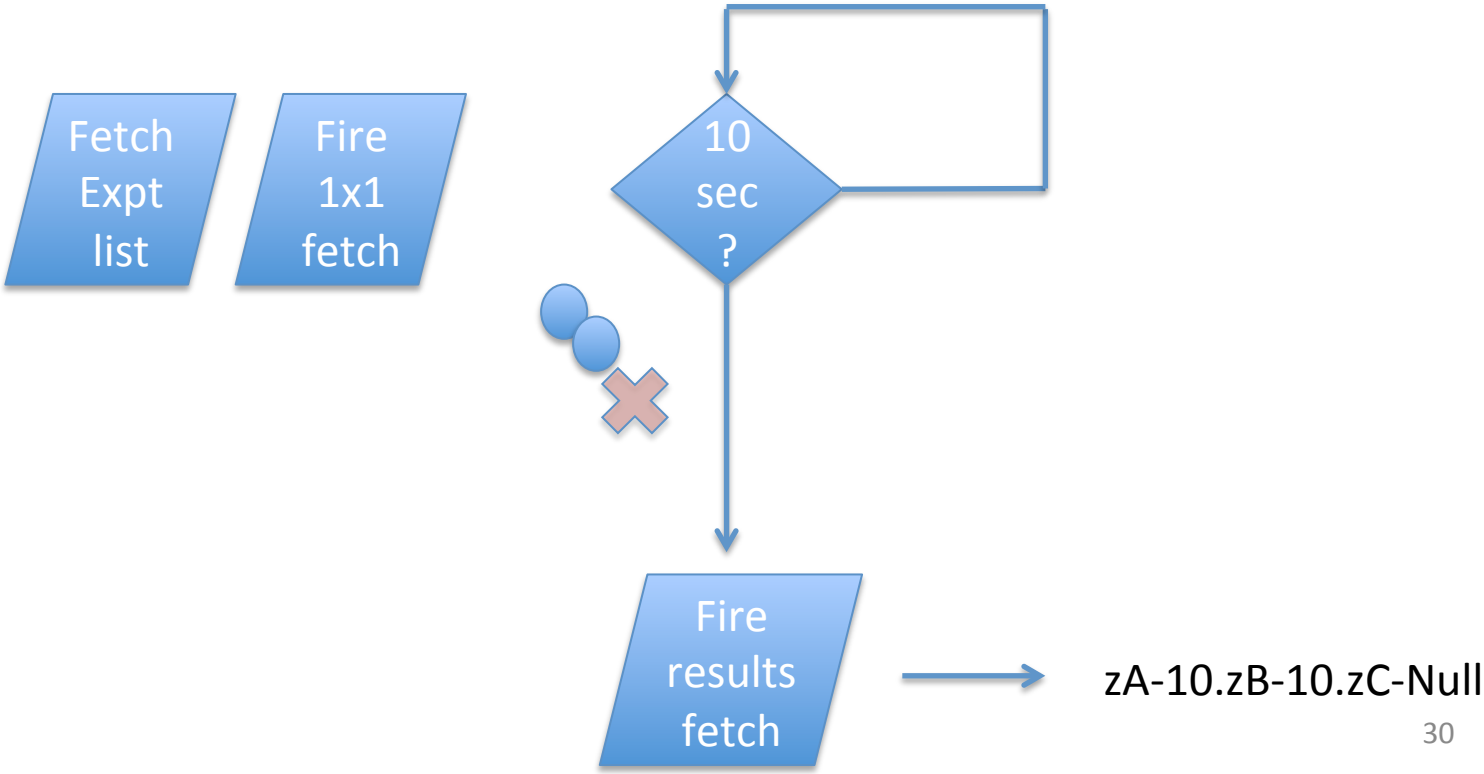
APNIC's ad

Thank you for helping us measure IPv6



APNIC's ad

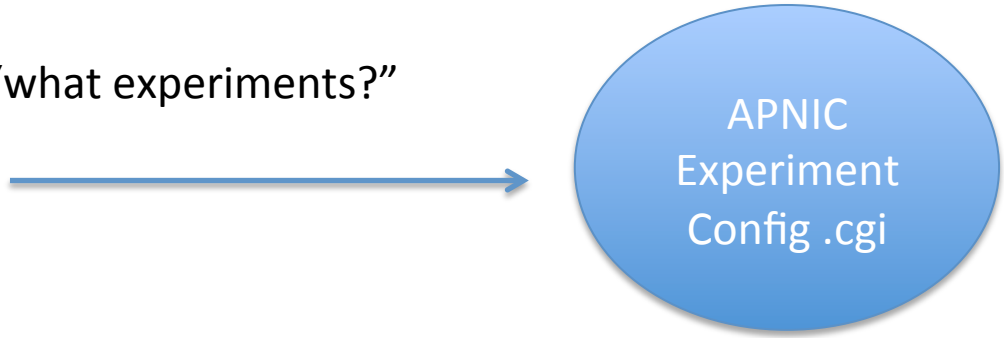
Thank you for helping us measure IPv6



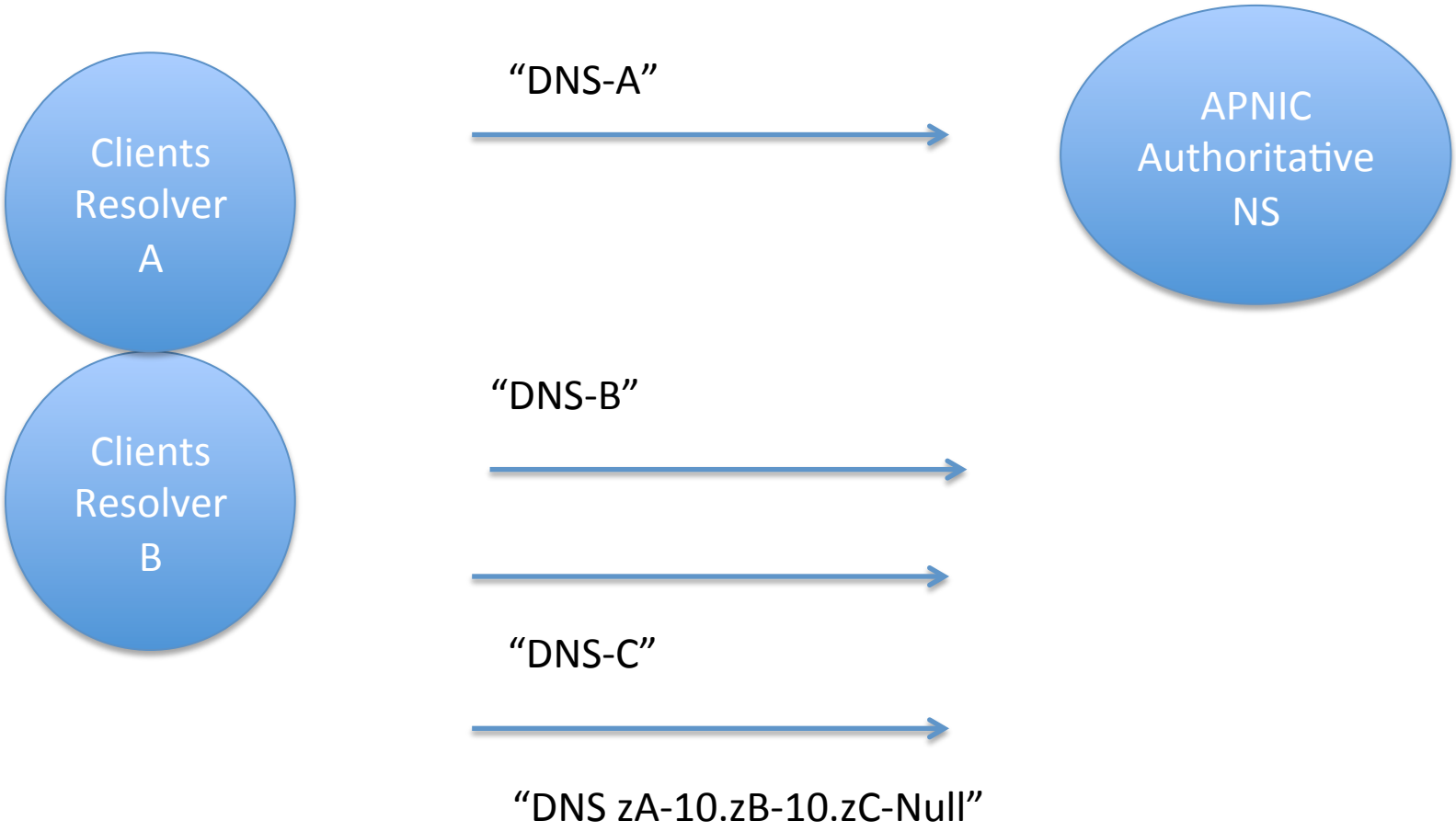
APNICs Server view

APNICs Server view

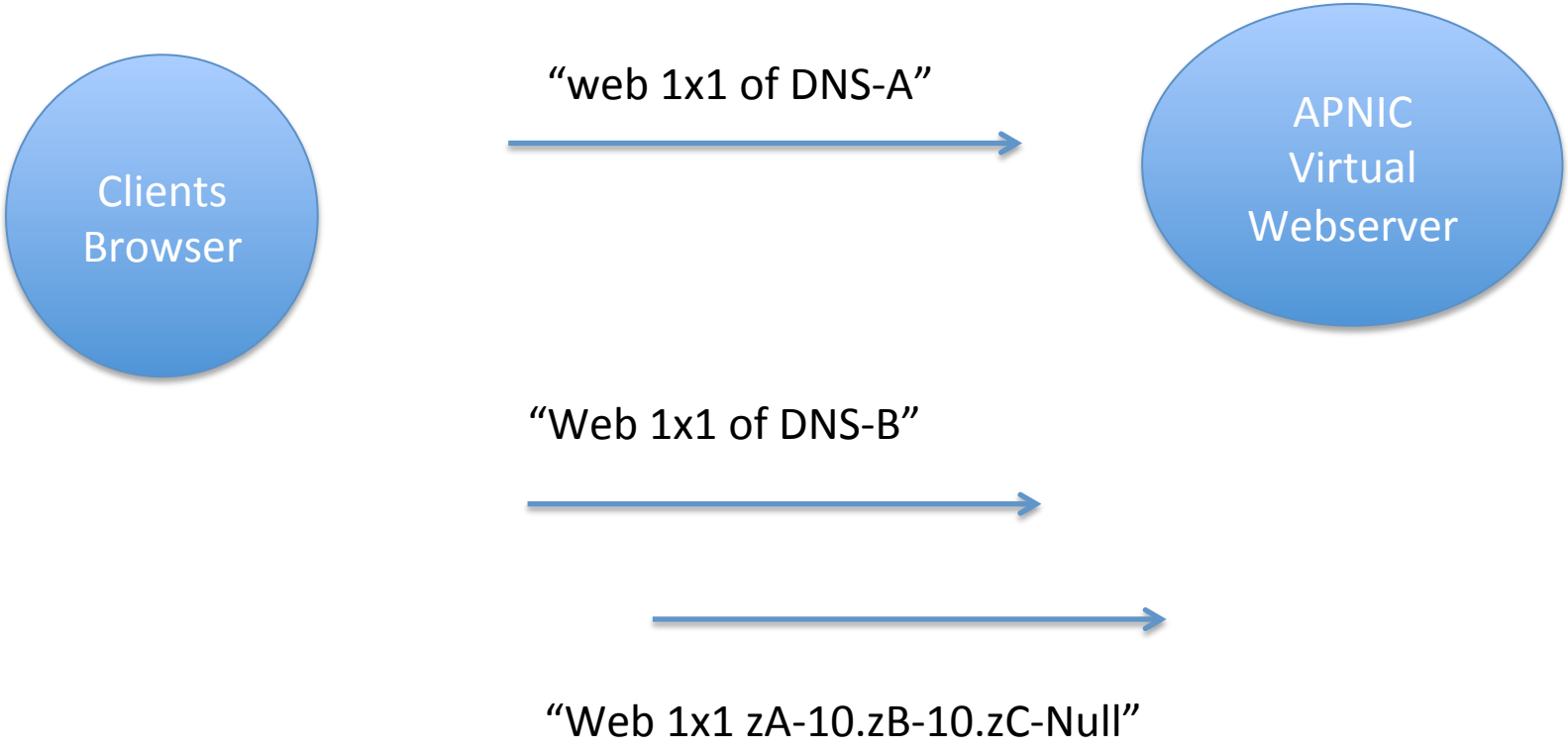
“what experiments?”



APNICs Server view



APNICs Server view



d http://z1.32c2d.z.dotnxdomain.net/1x1.png?d.t10000.u32c2d.s1366959402.i868.v6022.32c2d.z.dotnxdomain.net
e http://z1.32c2d.z.dashnxdomain.net/1x1.png?e.t10000.u32c2d.s1366959402.i868.v6022.32c2d.z.dashnxdomain.net
f http://z1.32c2e.z.dotnxdomain.net/1x1.png?f.t10000.u32c2e.s1366959402.i868.v6022.32c2d.z.dotnxdomain.net
results http://xr.x.rand.apnic.net/1x1.png?t10000.u32c2d.s1366959402.i767.v6022.32c2d&r=

d http://z1.32c2d.z.dotnxdomain.net/1x1.png?d.t10000.u32c2d.s1366959402.i868.v6022.32c2d.z.dotnxdomain.net
e http://z1.32c2d.z.dashnxdomain.net/1x1.png?e.t10000.u32c2d.s1366959402.i868.v6022.32c2d.z.dashnxdomain.net
f http://z1.32c2e.z.dotnxdomain.net/1x1.png?f.t10000.u32c2e.s1366959402.i868.v6022.32c2d.z.dotnxdomain.net
results http://xr.x.rand.apnic.net/1x1.png?t10000.u32c2d.s1366959402.i767.v6022.32c2d&r=

3 tests, and a results line. All three tests and results lie in experiment id ’32c2d’.

Two tests in dotnxdomain.net (DNSSEC enabled zone) and one in dash (no DNSSEC)

The malformed (bad DS in parent zone) is ’32c2e’

Data Collation

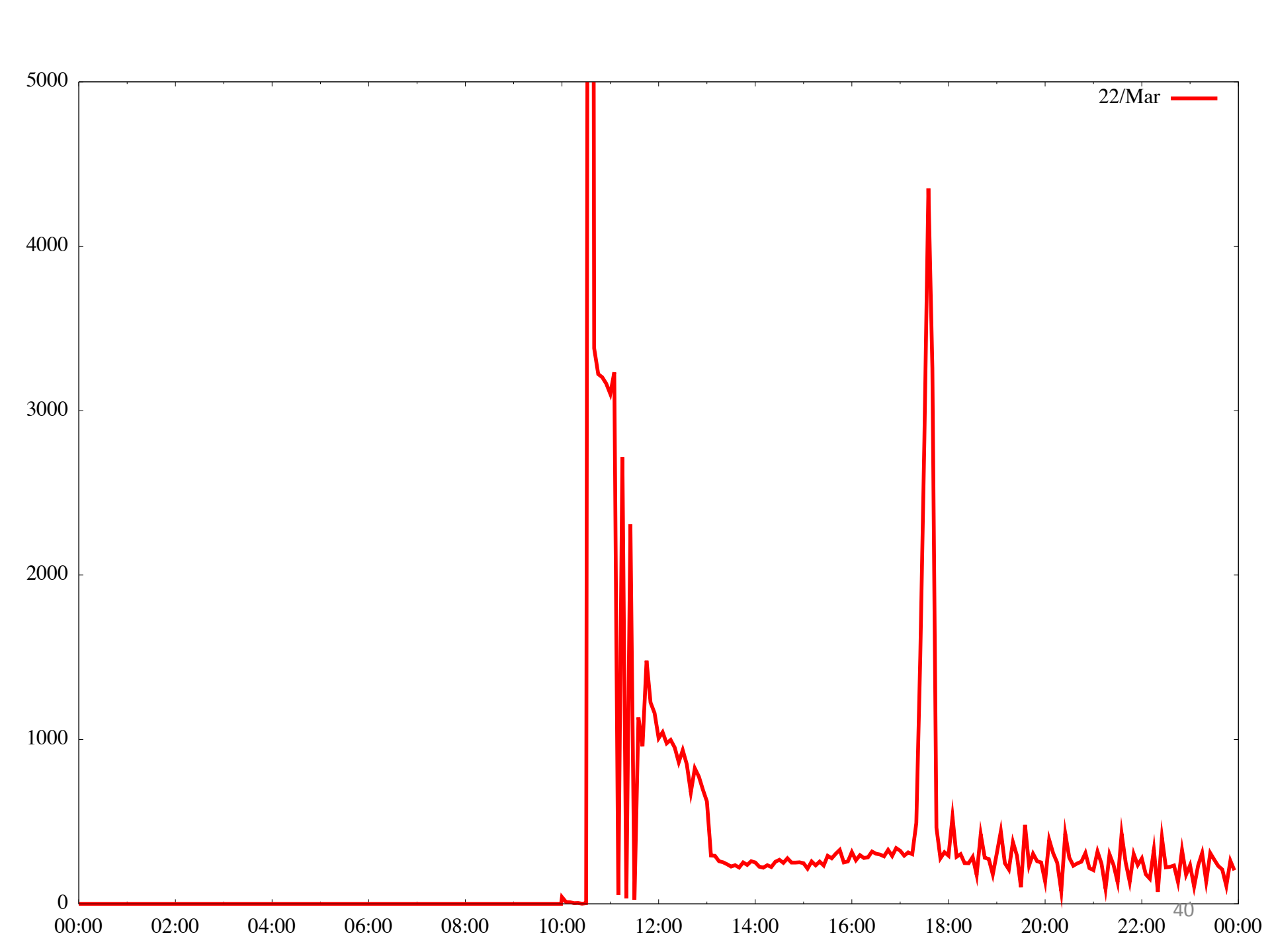
- All experiments given a unique timestamp and hash number by the head
- Collate DNS names, web fetches
  - Against client IP in web, resolver IP in DNS

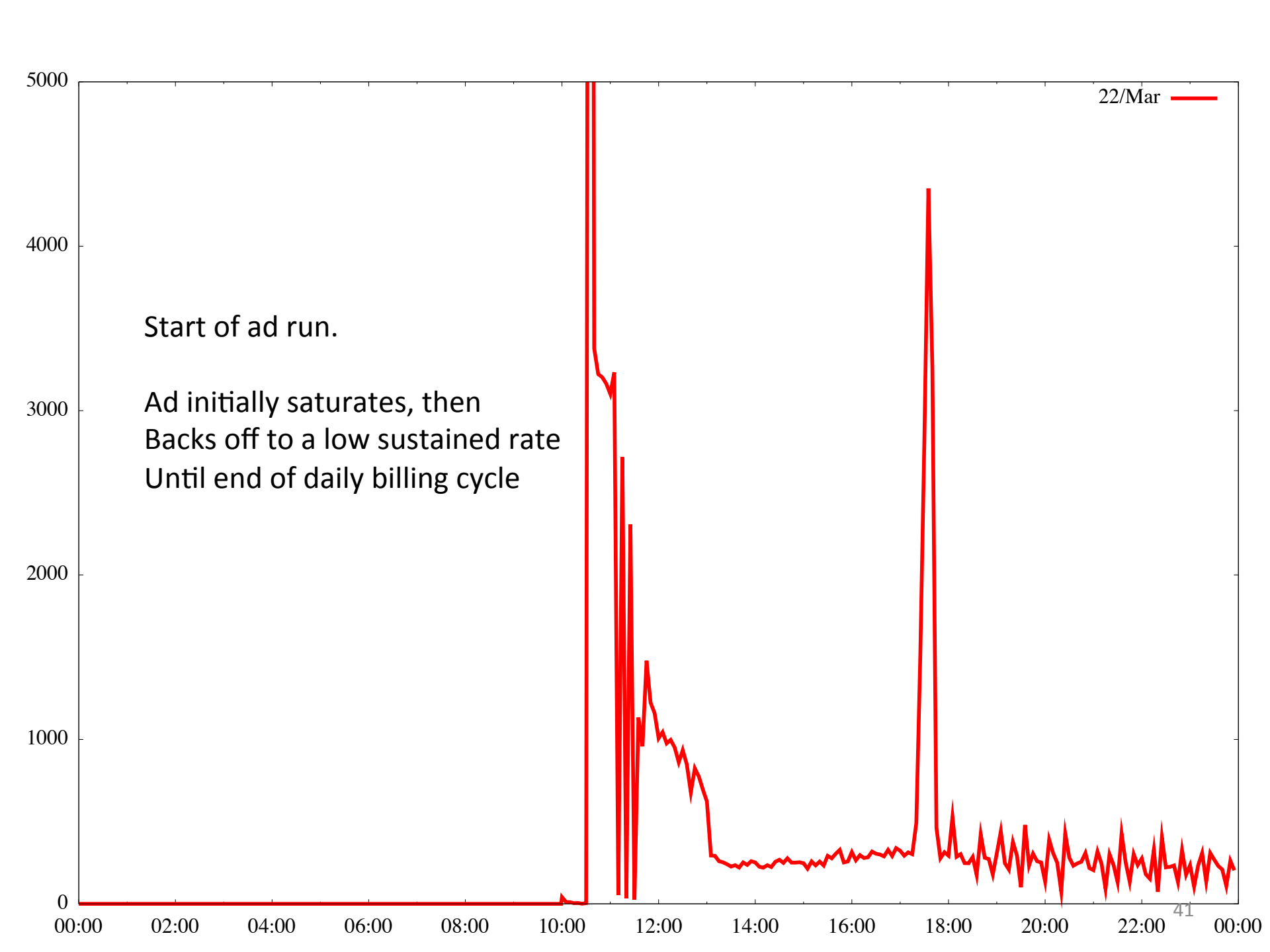
Advertising placement logic

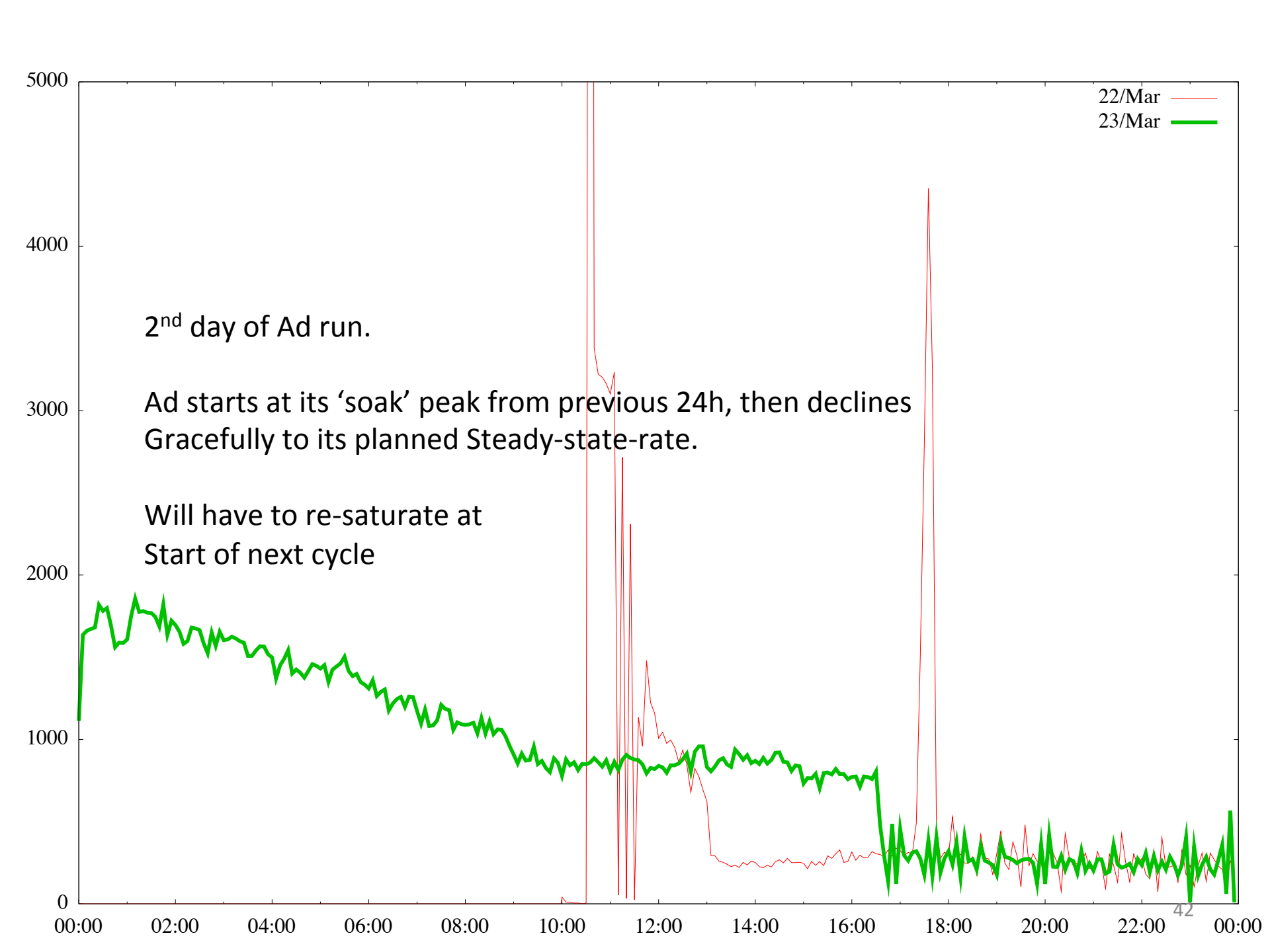
- fresh eyeballs == unique IPs
  - We have good evidence the advertising channel is able to sustain a constant supply of unique IP addresses
- Pay by click, or pay by impression
  - If you select a preference for impressions, then the channel tries hard to present your ad to as many unique IPs as possible
- Time/Location/Context tuned
  - Can select for time of day, physical location or keyword contexts (for search-related ads)
  - But if you don't select, then placement is generalized
- Aim to fill budget
  - If you request $100 of placement a day, then inside 24h algorithm tries hard to even placement but in the end, will 'soak' place your ad to achieve enough views, to bill you $100

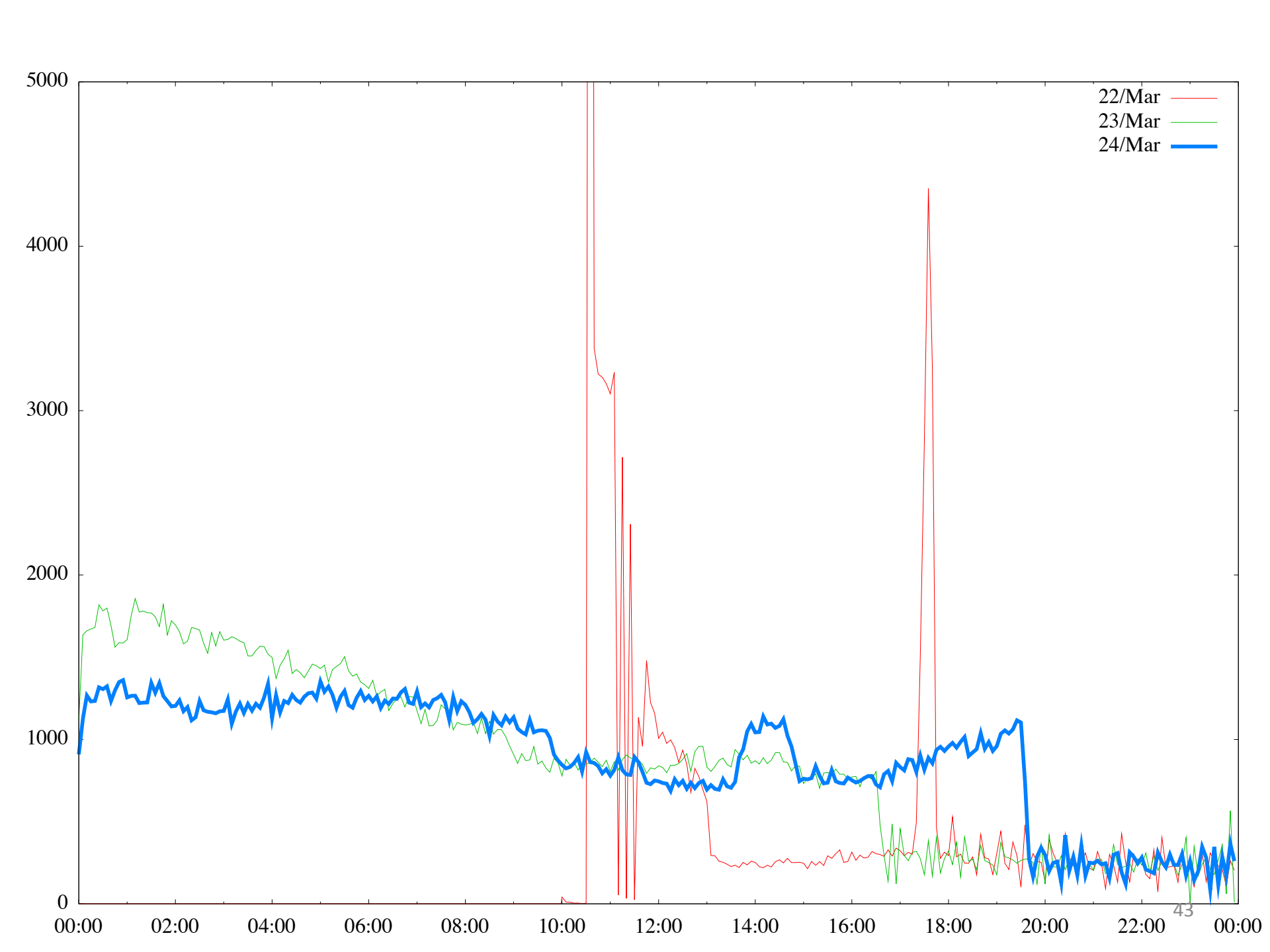
Advertizing placement logic

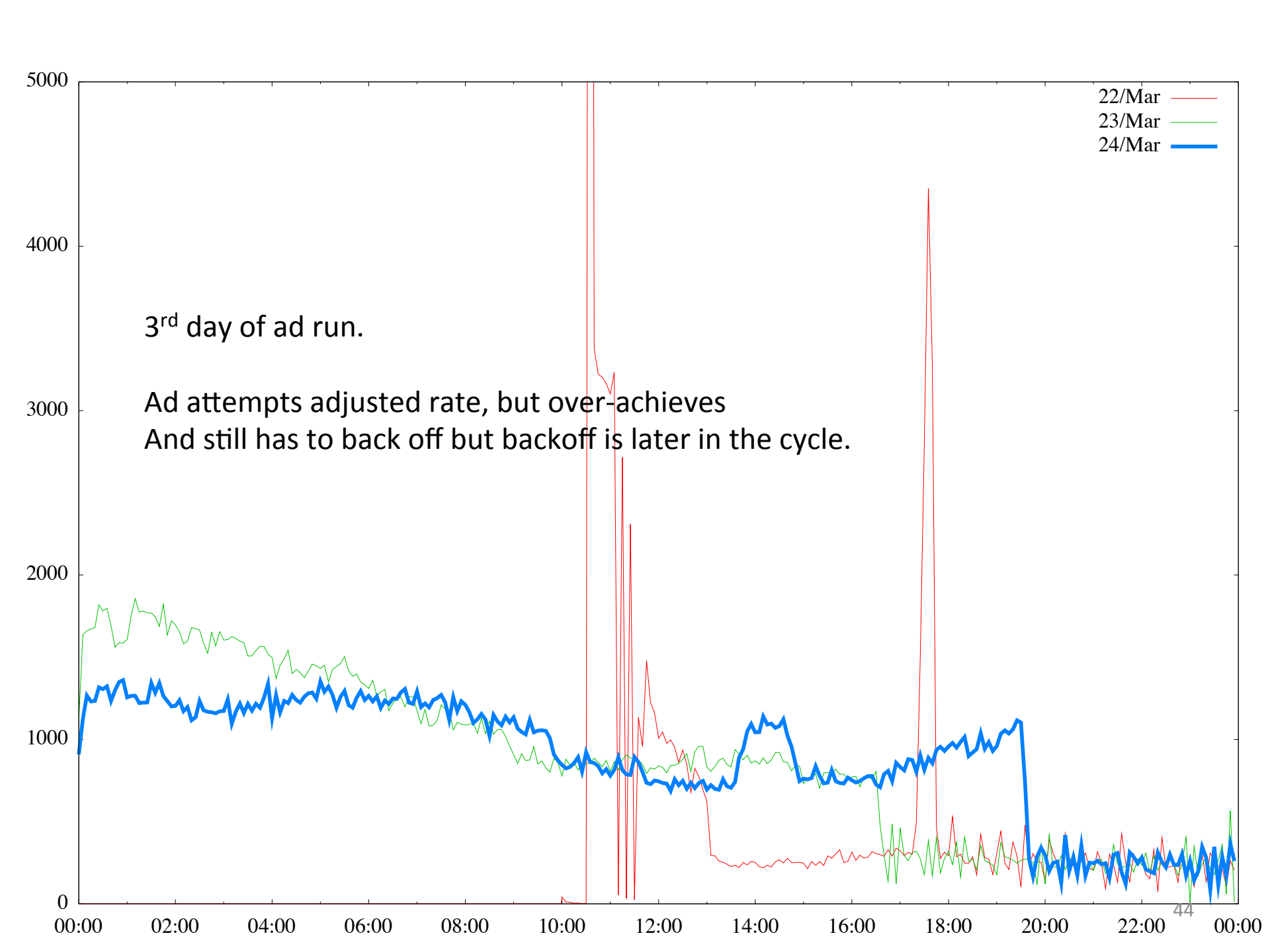
- Budget: $100 per day, at $1.00 ‘CPM’ max
  - Clicks per mille: aim to pay no more than $1 per click but pay up to $1 for a thousand impressions
- Even distribution of ads in the day
- No constraint on location, time
- Outcome: 350,000 placements per day, on a mostly even placement model with end of day ‘soak’ to achieve budget goal

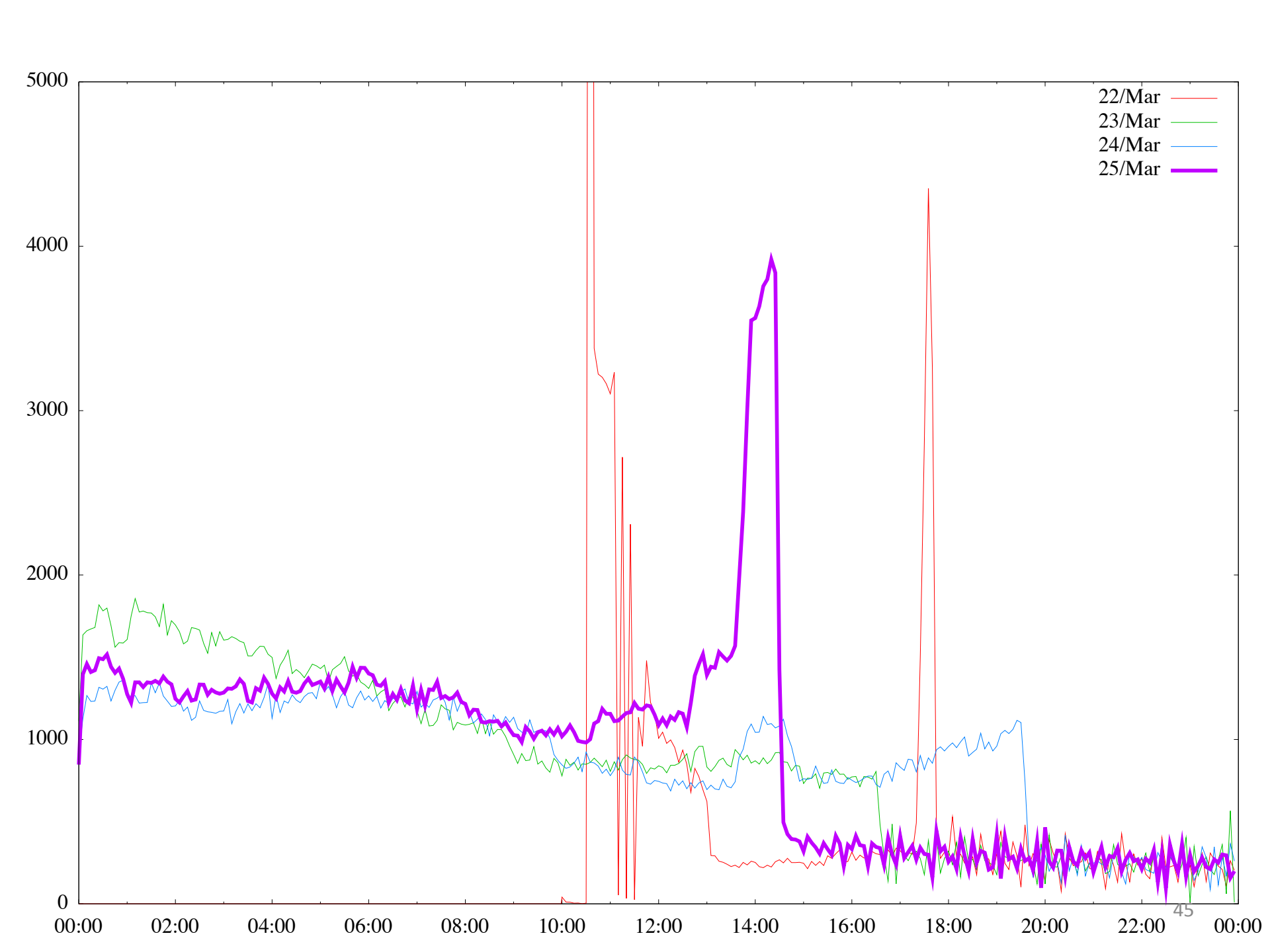

22/Mar
5000
4000
3000
2000
1000
0
00:00
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
20:00
22:00
00:00

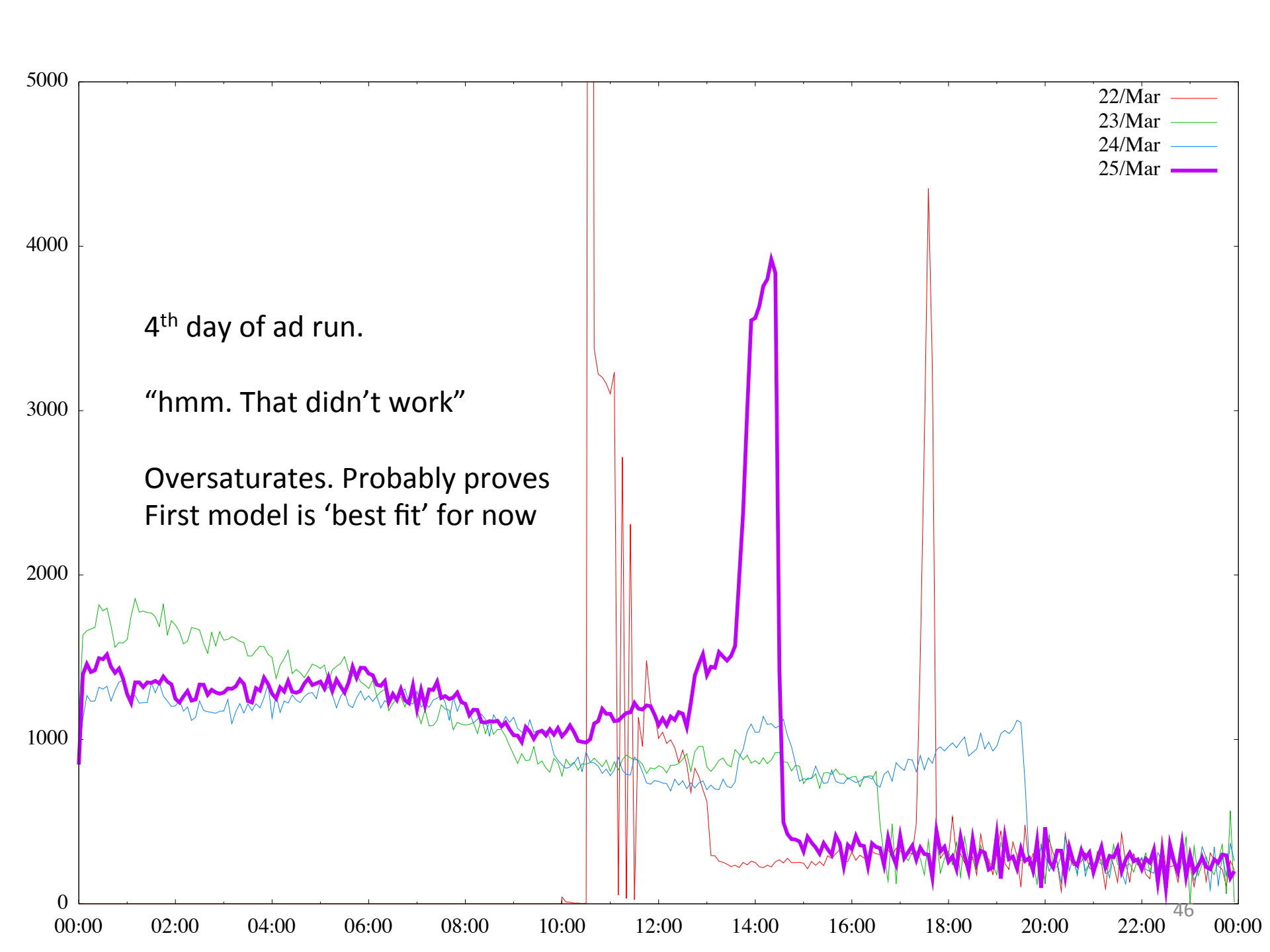

5000
4000
3000
2000
1000
0
00:00
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
20:00
22:00
00:00
22/Mar
Start of ad run.
Ad initially saturates, then
Backs off to a low sustained rate
Until end of daily billing cycle

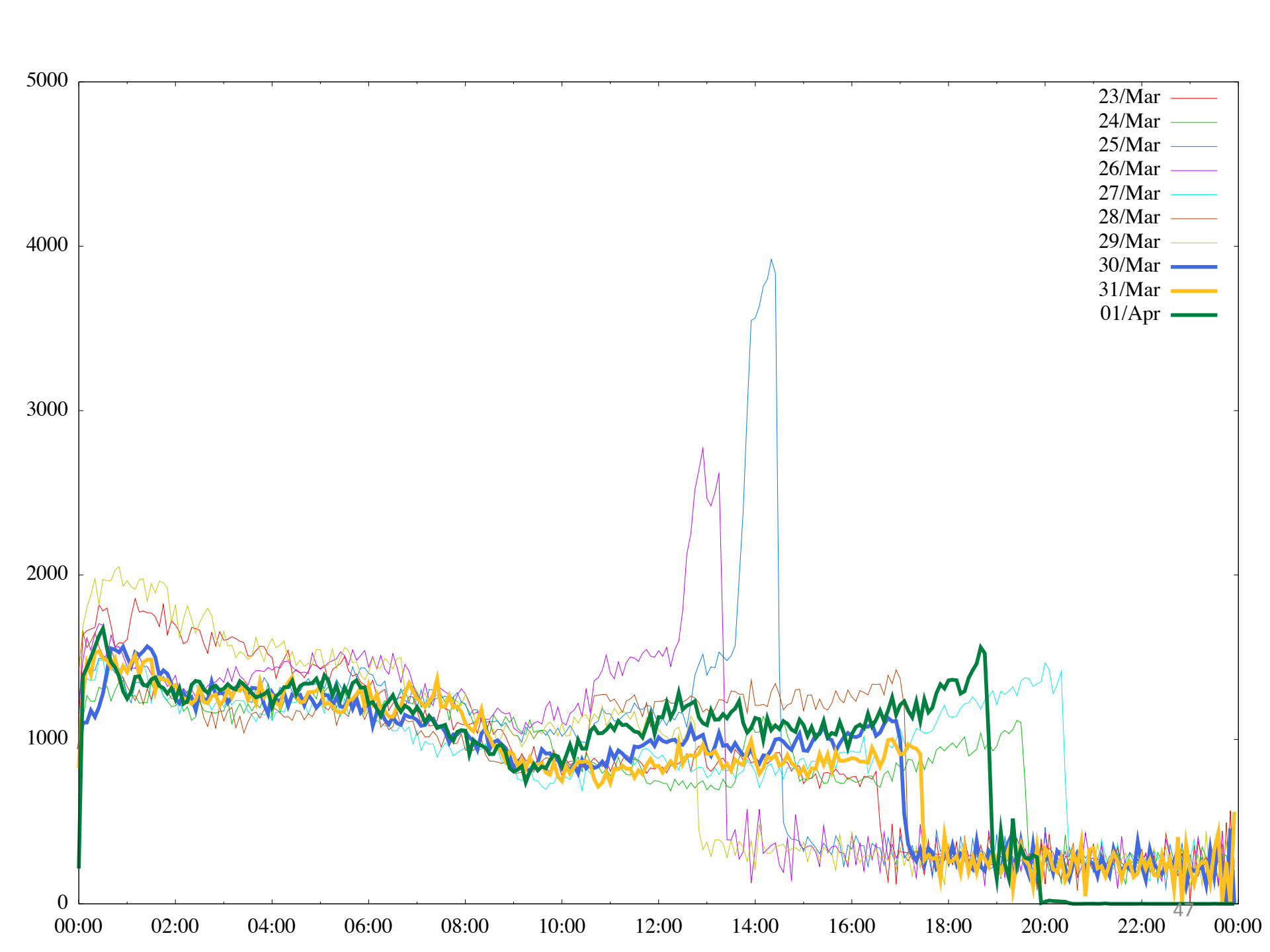

5000
4000
3000
2000
1000
0
00:00
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
20:00
22:00
00:00
22/Mar
23/Mar
2nd day of Ad run.
Ad starts at its 'soak' peak from previous 24h, then declines
Gracefully to its planned Steady-state-rate.
Will have to re-saturate at
Start of next cycle

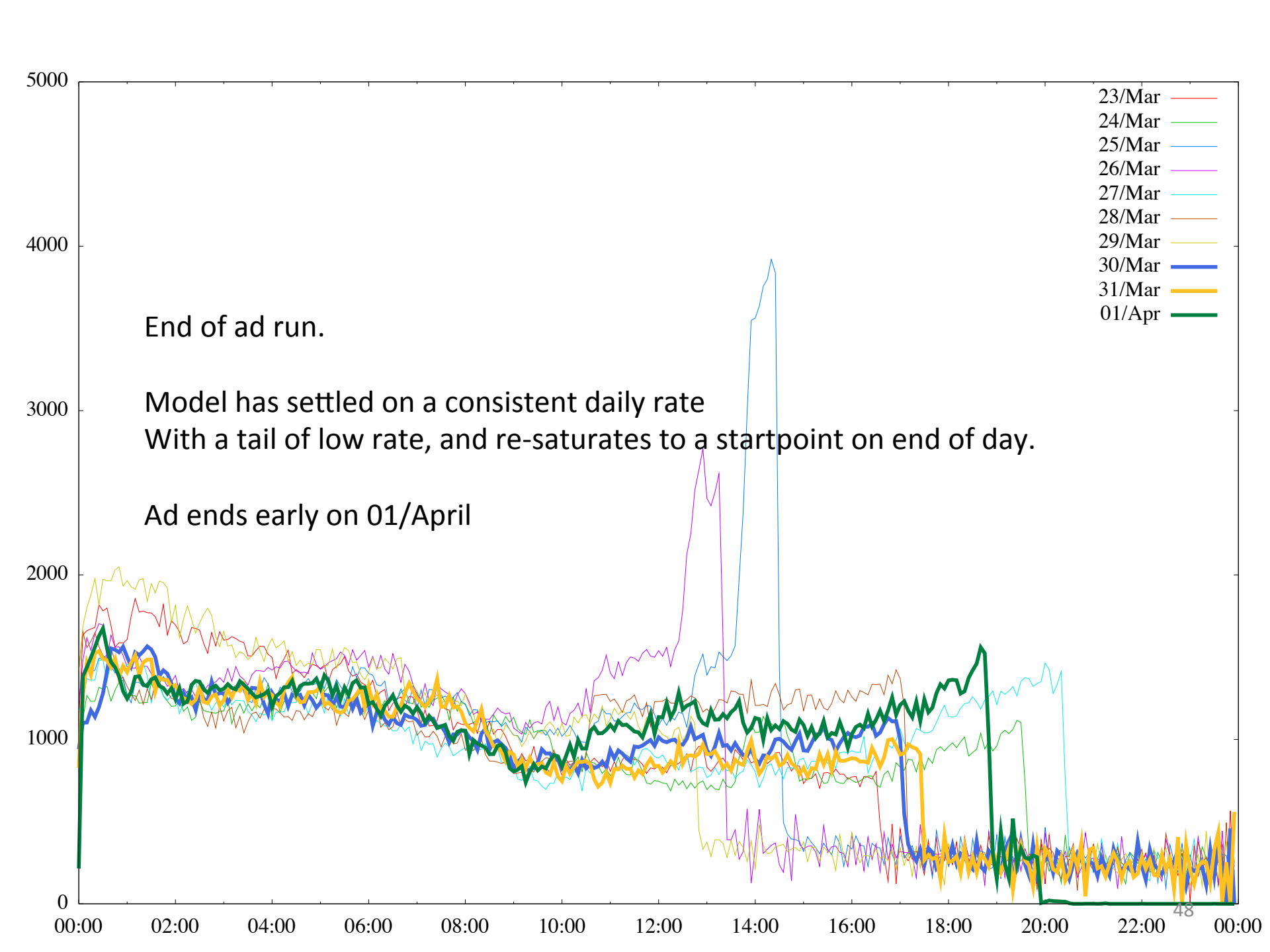

5000
4000
3000
2000
1000
0
00:00
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
20:00
22:00
00:00
22/Mar
23/Mar
24/Mar


5000
4000
3000
2000
1000
0
00:00
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
20:00
22:00
00:00
22/Mar
23/Mar
24/Mar
3rd day of ad run.
Ad attempts adjusted rate, but over-achieves
And still has to back off but backoff is later in the cycle.


5000
4000
3000
2000
1000
0
00:00
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
20:00
22:00
00:00
22/Mar
23/Mar
24/Mar
25/Mar


5000
4000
3000
2000
1000
0
00:00
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
20:00
22:00
00:00
22/Mar
23/Mar
24/Mar
25/Mar
4th day of ad run.
"hmm. That didn't work"
Oversaturates. Probably proves
First model is 'best fit' for now


5000
4000
3000
2000
1000
0
00:00
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
20:00
22:00
00:00
23/Mar
24/Mar
25/Mar
26/Mar
27/Mar
28/Mar
29/Mar
30/Mar
31/Mar
01/Apr


5000
4000
3000
2000
1000
0
00:00
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
20:00
22:00
00:00
23/Mar
24/Mar
25/Mar
26/Mar
27/Mar
28/Mar
29/Mar
30/Mar
31/Mar
01/Apr
End of ad run.
Model has settled on a consistent daily rate
With a tail of low rate, and re-saturates to a startpoint on end of day.
Ad ends early on 01/April

Experiment limitations

- No flash on most android
- Little or no flash placement on mobile device advertising channels
- Flash appears to serialize some network activity and even push()/pop() in reverse order
  - DNS and web serialization seen, re-ordering of fetches from a deterministic head serve from measurement controller
- Same technique possible in javascript
  - Harder to get good unique IPs from a specific website placement. A 1/10000 sample of wikipedia, or similar would be good…
  - (If you have a channel of a globally visible, popular website willing to embed .js we'd love to talk)

Experiments

- IPv6
  - Can the client use IPv6 only, dual-stack
  - Expose tunnels
  - (dns collected, but not yet subject to analysis)
- DNS
  - Can the client fetch resources where the DNS is
    - IPv6 enabled, IPv6 only
    - DNSSEC enabled, signed, invalidly signed
- Methodology looks applicable to other experiments. pMTU, IP reachability, HTTPS..

Generalized client/user experiment

- Model seems to permit a range of UDP,TCP, DNS and web to be tested
- Large number of worldwide footprint, or tuned delivery clients, random unique Ips
- Low TCO for datasets of order 5million
- Collecting long baseline data on deployment of resolvers in the global internet, and mapping of client networks to resolvers
- Sees large percentage of 8.8.8.8

DNS Experiments

- IPv6 DNS
  - Construct NS delegations which can only be resolved if the nameserver can fetch DNS over IPv6 transport
  - Explore pMTU/Tunnels by use of large DNS responses (2048 bit signatures, crafted hashnames which do not compress)
  - Does not text IPv6 reachability of client to web, explores IPv6 capability of DNS infrastructure

DNS Experiments

- DNSSEC
  - Construct NS delegations which have valid and invalid DNSSEC signed state, and see which clients appear to perform DNSSEC validation
  - And which fetch invalidly signed DNSSEC, even if validating (!)
  - Test depends on fetch of DS, DNSKEY to assert 'is doing DNSSEC'

NS delegation chain

- dotnxdomain.net managed at godaddy
  - Valid DNSSEC signatures uploaded
  - Passes public 'am I dnssec enabled' checkers on web
- z.dotnxdomain.net validly signed subdomain
  - XYZAB.z.dotnxdomain.net subdomains
    - Half (even) signed invalidly
    - Half (odd) signed validly (invalid DS in parent)
- Matching dashnxdomain.net (no DNSSEC)
- 250,000 hex-encoded stringnames
  - 435Mb zonefile of sig chain over subzones. 30min load time.

Why 250,000?

- Measurement of advertising placement rate indicated under 125,000 ads/ hour was peak seen (70,000) at our budget
- Ensures that a complete cycle of all unique odd or even experiment subdomains cannot exhaust (recycle % 250000 modulus) inside zone TTL of 1hour
- Therefore ensures that every experiment served lies outside any resolvers cache
  - Noting that some resolvers re-write TTL
- Therefore DS/DNSKEY for parent of any test is not cached, and we therefore 'see' parent DNSSEC fetch associated with experiments
- No DS/DNSKEY of parent inside <short window> == no DNSSEC
- Fetch of web asset correlated by wildcard name having unique time, serialcode embedded in DNS, web which match. Absence of fetch of invalidly DNSSEC signed web asset used as signature 'validation outcome obeyed'

Why intermediate domains?

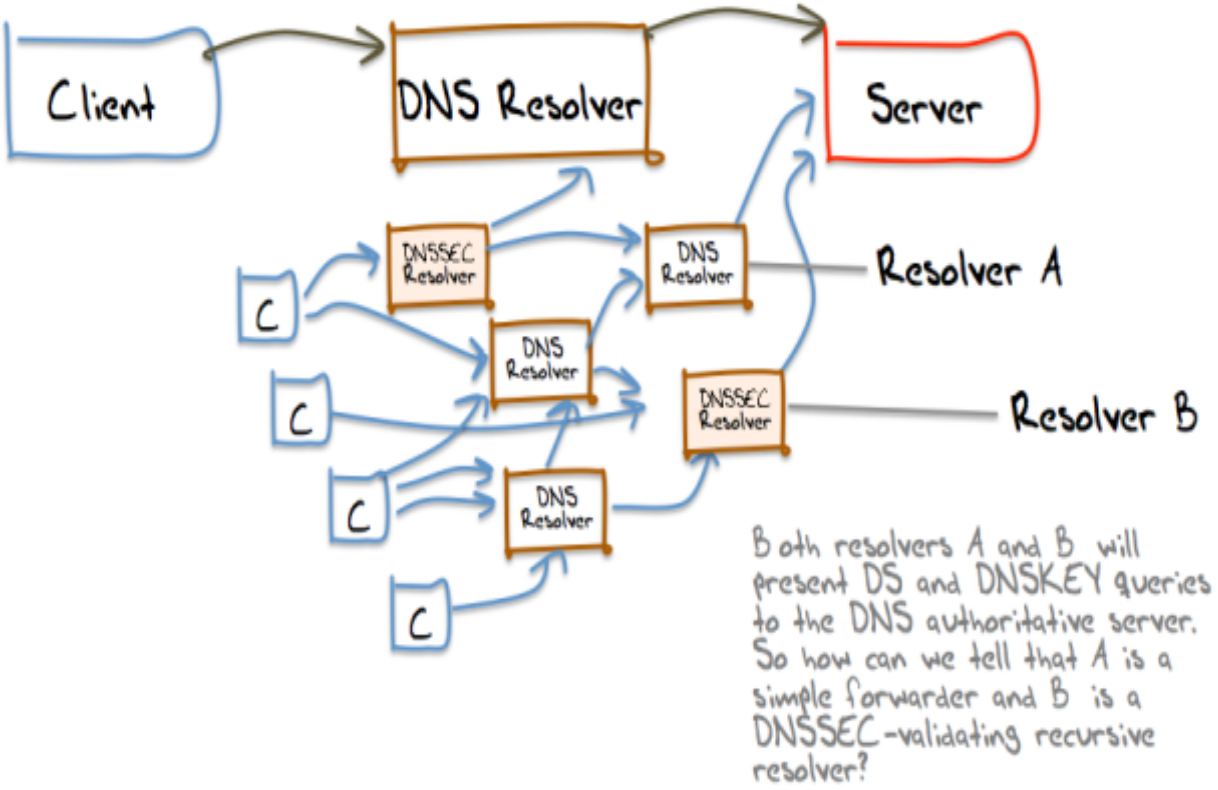
- Dotnxdomain.net signed under .net
- .z subdomain forces into independent NS chain we can understand has single NS
- .xxxxx.z. subdomain(s) force NS for a domain which is not currently in cache
  - NS is *not* the same IP as the z.subdomain therefore no short-circuit knowledge possible in the answer.
  - DS and DNSKEY must be seen on single NS to be performing DNSSEC. Once cached, may then not be tested by DNSSEC aware resolver until ttl timeout.
- *.xxxxx.z. wildcard inside xxxxx subdomain serves any name under the domain
- Virtual server name logged in web, visible in tcpdump of GET

DNS is complicated

DNS is complicated

DNS is complicated

How can we interpret what we are seeing?



What we see

- We see the head (last) resolver coming to the authority NS
  - Only one NS, so all traffic comes to us
- The 'head' is now often multi-ip
  - Google 8.8.8.8 backed off a farm
  - A from one server, AAAA from another
  - TCP and UDP from different servers
- 'rational' Query order not preserved
  - (A followed by DS/DNSKEY..)

Our working definition of ‘does’ DNSSEC

- IF within a time limit we see:
  - A of the terminal
  - DS, DNSKEY of the parent
- AND
- IF the web fetch doesn’t go to a ‘bad’ DNSSEC
  - The f experiment
  - AKA ‘won’t follow a lie’
- Looks to us like DNSSEC enabled

Our working definition of ‘not doing DNSSEC’

- Never does DS/DNSKEY queries
- Goes to f, since never sees DNSSEC state
- Which leaves..
  - A bunch of people who do ‘some’ but not all

Our working definition of ‘not doing DNSSEC’

- Never does DS/DNSKEY queries
- Goes to f, since never sees DNSSEC state
- Which leaves..
  - A bunch of people who do ‘some’ but not all
- DNSSEC fail == “servfail”
  - Try next /etc/resolve.conf nameserver entry
  - No DNSSEC? See everything! (goes to f)

Does/Doesn't do DNSSEC

Used DNSSEC Validating Resolvers:	64,690	3.35%
Used a mix of validating and non-validating resolvers:	43,657	2.26%
Fetched DNSSEC RRs some of the time:	2,652	0.11%
Did not fetch any DNSSEC RRs:	1,816,968	94.13%

Look on the bright side

Look on the bright side

- This is better than worldwide IPv6 uptake

Google's DNS(sec)

- At time of experiment, no strong evidence of systematic DNSSEC from google 8.8.8.8
  - Appeared to be partially DNSSEC: acted as a cacheing DNSSEC aware resolver, not fronting for clients with no DNSSEC.
  - Some evidence of interpreted behaviour with client set CD/AD flags
  - But.. They just made more changes. …

Who uses google?

- 2,696,852 experiment ids (march)
  - In principle, one per unique client
  - Some re-use from caches/cgn/proxies..
- 197,886 used google backed DNS
  - 7.33% of total population using google DNS
  - 125,144 used google **EXCLUSIVELY**
    - 63% of google use appears exclusive
    - 4.64% of total population using google exclusively

About those changes..

- First pass analysis of 4 days data

About those google changes..

- First pass analysis of 4 days data
  - 514,000 experiments with completed results
  - Excludes people who dropped out before 10s
    - Not using DNSSEC 438,287 85.26%
    - Partially DNSSEC 32,976 6.41%
    - Fully Validating 42,737 8.31%
    - Using google: 38,950 7.57%
      - Fully validating, using google: 22,336 52.26%
      - Fully validating, not using google: 20,401 47.73%
- Fully validating now ~ 8.31% 52.26% from google

Top 25 DNSSEC economies

#econ	total	% google	% dnssec	% google dnssec	% not google dnssec
SE	1363	4.26	81.29	1.99	98.01
SI	1106	6.60	56.51	8.00	92.00
VN	4929	40.23	36.40	98.49	1.51
CZ	8100	9.89	32.81	15.27	84.73
FI	572	1.05	32.17	2.17	97.83
CL	7936	2.66	31.93	4.06	95.94
IE	1881	3.56	27.96	7.60	92.40
ID	10361	22.09	23.04	77.96	22.04
UA	5365	15.86	20.63	23.40	76.60
EG	6450	25.69	19.15	99.76	0.24
US	29361	3.10	18.71	7.28	92.72
TR	12268	19.39	17.70	96.82	3.18
AZ	1517	51.55	14.63	100.00	0.00
PE	1930	15.54	14.04	87.82	12.18
BR	34738	11.61	13.34	55.61	44.39
IT	19794	14.22	13.05	99.19	0.81
DZ	1875	39.15	12.59	99.15	0.85
EC	1719	7.62	12.45	35.05	64.95
PK	1912	20.45	11.77	96.00	4.00
NO	896	1.45	11.16	6.00	94.00
MY	7802	12.66	9.61	95.73	4.27
PS	591	55.50	9.14	100.00	0.00
AL	1309	15.74	8.40	100.00	0.00
IQ	1446	22.89	7.95	71.30	28.70
AM	1376	10.47	7.70	92.45	7.55
PH	13134	6.70	7.44	44.32	55.68

Oddies

- Most back-end resolvers prize goes to…

Oddies

- Most back-end resolvers prize goes to…
  - An anonymous comcast user with 56 resolver IPS
    - Most inside 8.8.8.8 backend nets
    - Still managed DNSSEC despite the SERVFAIL chain from hell
- There was one with 60 but didn't do DNSSEC
  - Also using 8.8.8.8 … but also NOT using 8.8.8.8
    - Once. (its all it takes)
- Four(ish) queries with AD bit set. In 5million
  - What was that about deployment time for new bind versions?

Time for a RANT

Time for a RANT

- SERVFAIL != DNSSEC FAIL

Time for a RANT

- SERVFAIL != DNSSEC FAIL
- >1 resolver in /etc/resolv.conf
  - One DNSSEC, one not
  - Equivalent to “no DNSSEC” protection
- If one backed by 8.8.8.8 then same outcome

SERVFAIL != DNSSEC FAIL

- Great during transition, Not much help in a modern world
- Maybe some richer signalling inside Additional?
  - “stale SIG at parent”
  - “no DS you twonk”
  - “bad RRSIG somebody’s fingers in the zone”
  - “TA mismatch you are not RFC5011 goodbye”

How much 5011

- DNSSEC keyroll inline signalling comes in rfc5011
  - Late 2009.

How much 5011

- DNSSEC keyroll inline signalling comes in rfc5011
  - Late 2009.
- How much of the world runs bind 9.7 or newer?

How much 5011

- DNSSEC keyroll inline signalling comes in rfc5011
  - Late 2009.
- How much of the world runs bind 9.7 or newer?
  - “we don’t know, because we don’t have a signal of resolver version in the query packet”

No signalling?

- You can’t do DNSSEC without EDNS0

No signalling?

- You can’t do DNSSEC without EDNS0
- So we have to be in extended DNS capabilities to do DNSSEC DO ok…

No signalling?

- You can't do DNSSEC without EDNS0
- So we have to be in extended DNS capabilities to do DNSSEC DO ok…
  - So why not put some additional in to EDNS0 enabled resolvers to signal what they can do?

No signalling?

- You can't do DNSSEC without EDNS0
- So we have to be in extended DNS capabilities to do DNSSEC DO ok…
  - So why not put some additional in to EDNS0 enabled resolvers to signal what they can do?
    - p0f is just a model.
    - There is low information leakage risks
    - We'd get some sense of what the length of the long tail is

That long tail…

- ‘stuck’ DNS queries
- 4 nodes seen doing >1000 repeated queries in DNS for the same label
  - Low rate 1/sec background noise
  - But..

That long tail…

- ‘stuck’ DNS queries
- 4 nodes seen doing >1000 repeated queries in DNS for the same label
  - Low rate 1/sec background noise
  - But..
  - This is a bug in a bind-4 release.
  - So we know we saw at least 4 instances of a bind4 node.
  - So how long is that long tail of non-RFC5011 resolvers?

We tested one NS zones

We tested one NS zones

- If you have 4 NS
  - Bind tests all 4 NS, if the zone is mis-signed
- If you have 11 NS
  - Bind tests all 11 NS, if the zone is mis-signed
- No cached state. Re-tests each time asked
- Doesn't combine if 2 levels broken
  - Terminates at highest break in DNS tree
- However: there is an explosion of traffic on the authoritative server from broken DNSSEC
- And its large(r) packet output

Not Google!

- Google doesn’t do this
- Google limits to ?one? NS test, if validly signed itself, doesn’t requery alternates if chain broken

TA rolling and rfc5011

- If we roll the TA, and if resolvers have hand-installed trust, and don't implement 5011 signalling
- How many will say "broken DNSSEC" when the old sigs expire?
- How many will re-query per NS high in the tree to the authoritative servers?
- What percentage of 3%->4% of worldwide DNSSEC will do this?

..google..

- If the model is to move to 8.8.8.8 then most DNSSEC growth is coming from a query limiting source, which is rfc5011 aware
- SERVFAIL mapping and >1 resolver means most people failover to a non-DNSSEC resolver anyway
- Whats going to happen if we roll TA isn't yet clear…
  - MORE QUESTIONS NEED TO BE ASKED

Conclusion

- Experimental technique for performing DNS, web Experiments
- Good source of worldwide random client ips
- ~2,000,000 tests suggest 3.5% DNSSEC deployment (more data to come post google announcement of complete DNSSEC)
- Increasing use of google by end users
  - Including exclusive use of google DNS
- There are lots of questions… Can we help answer any of them?
  - Large dataset of worldwide mapping client:resolver
  - Not ‘open’ DNS based: reflects real-world client use

More details at…

- http://labs.apnic.net/blabs/?p=341
  - Measuring DNSSEC performance
- http://labs.apnic.net/blabs/?p=316
  - DNSSEC and Google's public DNS service