Some Notes on the Modern Web

Mark Stapp, cisco systems
icnrg at IETF 89, London, March 2014

We've been spending some time looking at a few popular/familiar websites, trying to understand what they look like currently in tcp/http terms. We're starting to explore what the modern web might look like in a future NDN/ICN world.

Sampling Sites

- Sampled home pages of 16 familiar/popular sites
  - Included portal, commerce, brick-and-mortar, content
- Used Chrome dev tools to capture http archive (har) logs
  - Clear browser cache, clear cookies, no ad-blockers, script-blockers or other extensions enabled
- For now, this is mainly descriptive (and not fabulously scientific.) Our goal was to see whether there were some interesting characteristics or relationships in modern websites worth exploring further from the ICN/NDN perspective.

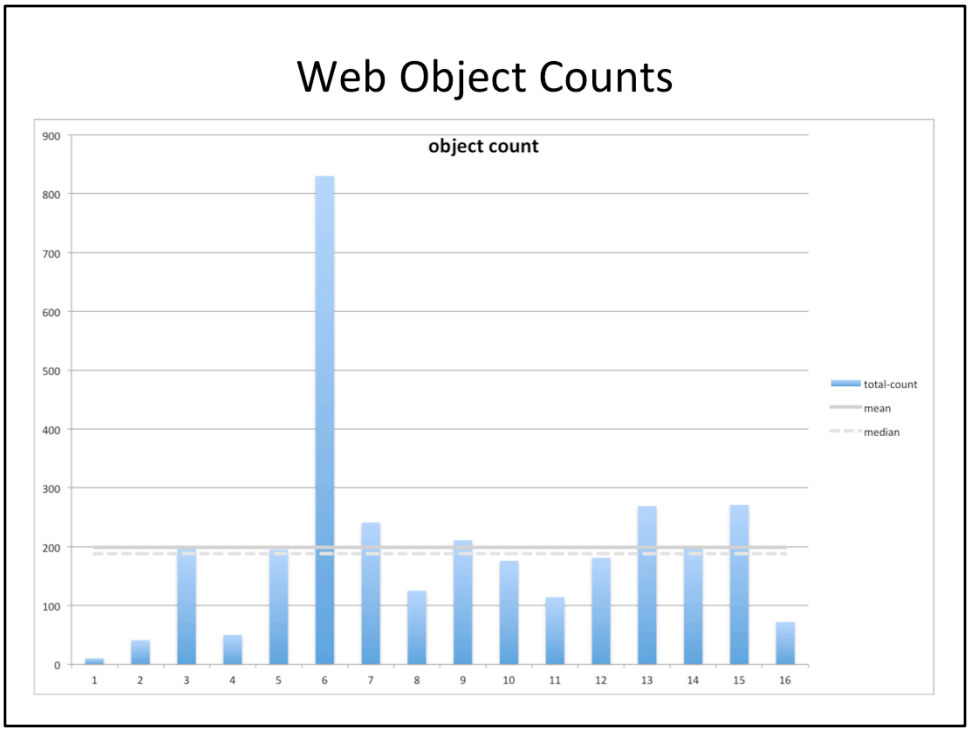
Some notes on the tools we used to capture information from the sites we sampled.

Web Sites are Complicated, RESTful, and Commercial

- In our initial sample:
  - There's a range, but there are a couple of hundred objects on each page, on average
  - Objects come from a couple of dozen DNS domains
  - Many / most of the objects are small (< 4KB)
  - Most requests are 300+ bytes: these are RESTful requests
  - Significant fraction of requests are >= response object size
  - Most sites use 3rd party tracking, metrics, and frameworks
- Which characteristics have implications for ICN/NDN? What is easier/more efficient in NDN, and what is challenging?
- We'll model some possible approaches and offer those results (eventually)

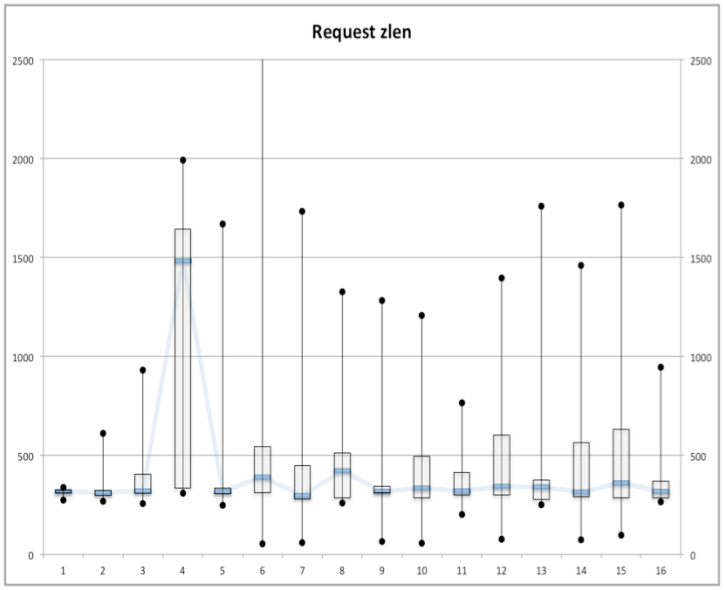
These pages don't fit a simple "fetch the html page" model. They contain hundreds of objects, mostly small ones, requiring many tcp connections. Those connections don't just reach 'the server' (or 'the publisher') – there's no such thing; there are dozens of destination servers in CDNs, ad networks, metric/tracker networks.

Modern browsers have evolved to try to reduce the impact of this with aggressive local caching of content, dns data, and parallel, 'reuseable', or even 'speculative' tcp connections. The transports - http (via pipelining and SPDY, e.g.) and tcp (via iw=10, e.g.) - have been stretching to adapt also. Fascinating (ymmv) QUIC from google goes even further, removing tcp and using udp to carry http.

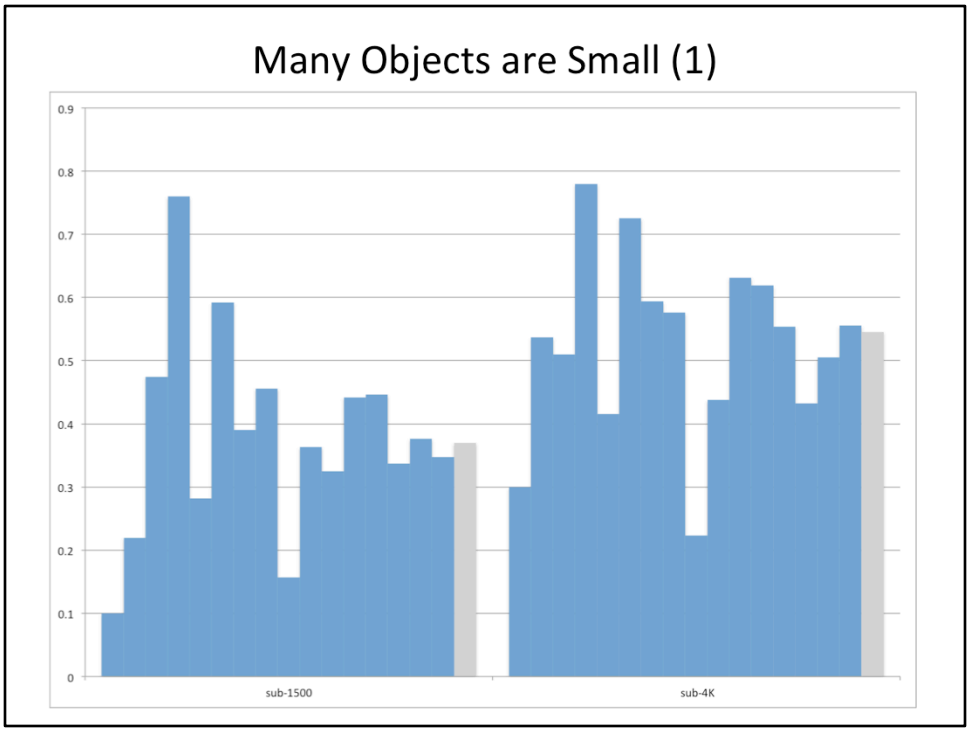


Unique requests/objects on each home-page, with the mean and median superimposed as horizontal lines. There are some outliers here, but the point is that ‘a page’ is not ‘an object’, it’s a big complex of dozens or hundreds of objects. FYI, column number 1 is boston.craigslist, which is very plain, just a bunch of links.

Request Sizes



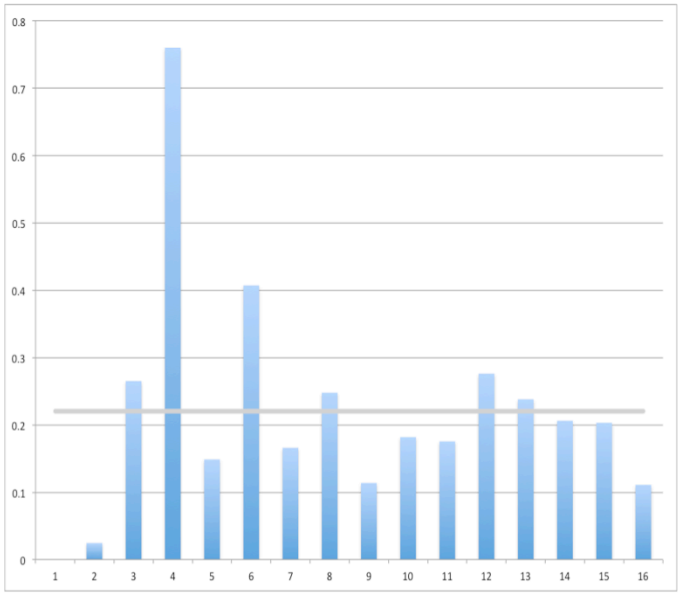
The high-low lines show the range of (gzipped) request sizes on each page. The bars show the 25% and 75% points, and the line shows the medians.
Again, there are some outliers, but most requests carry hundreds of bytes of information.



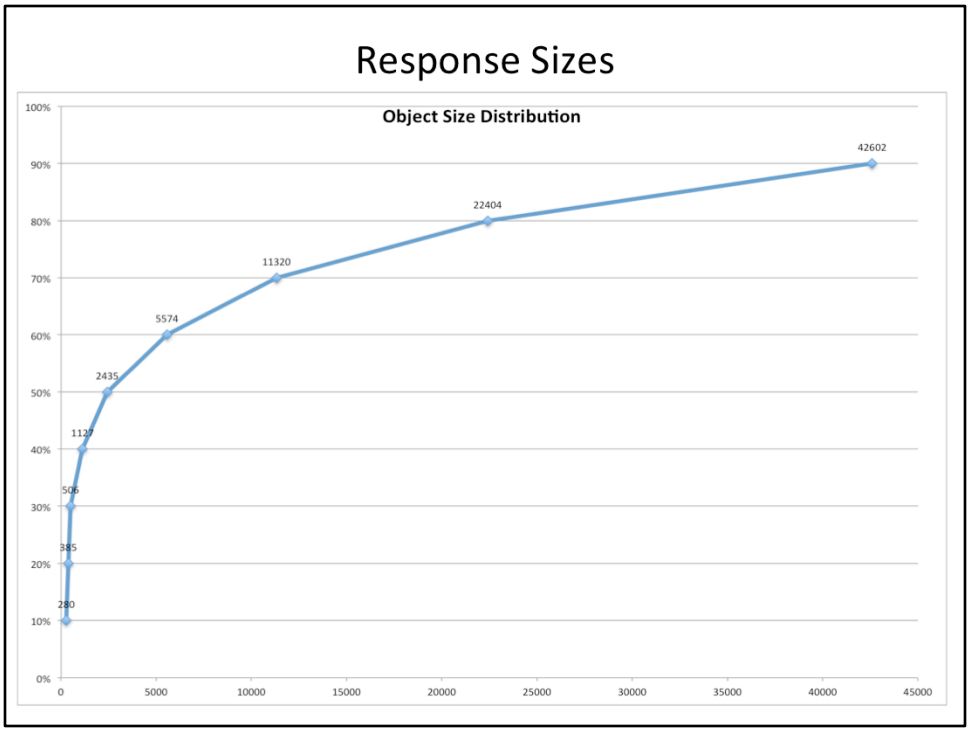
These are the percentages of objects on each site that are <= 1500 bytes and <= 4K (more-or-less one Content message.)
(The mean of each group is in the right-most column in grey.)

The point here is that a significant fraction of the objects are quite small, small enough to fit in a single “Content” message at the canonical 4K size.
Interestingly, almost 20% of the objects are smaller than the request sent – that is, the client has to send more data than it receives in those cases.

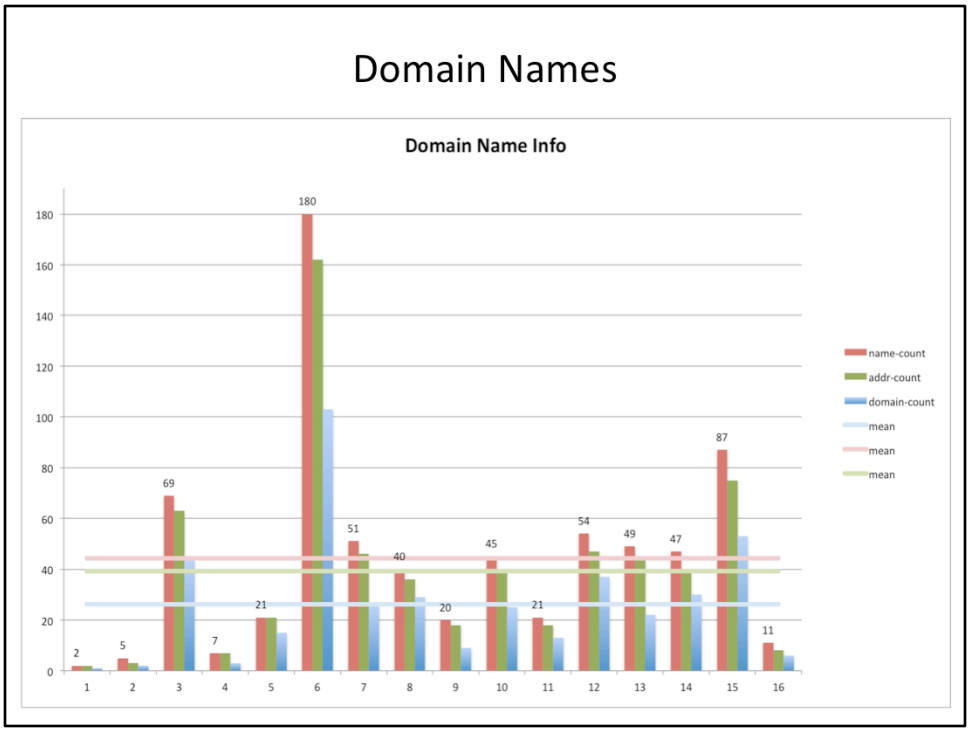
Many Objects are Small (2)



This shows the percentage of responses that were <= the corresponding request. The average is overlaid as the grey horizontal line.
Interestingly, over 20% of the response objects are smaller than the request sent. That is, the client had to send more data than it received in those cases.



This is sort of another way of looking at response object sizes. Along the lines of a cdf plot, this graphs the distribution of object sizes at 10% intervals. The 50% point is just under 2500 bytes, and the 90% point is just over 42KB.



These are counts of the unique url location names (in red) present on each homepage. In http, these represent DNS names that have to be resolved. The green bars are the number of unique ipv4 addresses associated with those names: each address has to be the target of a tcp connection. On average, dozens of connections are required to retrieve the entire page. The paler horizontal lines are the averages for each category.

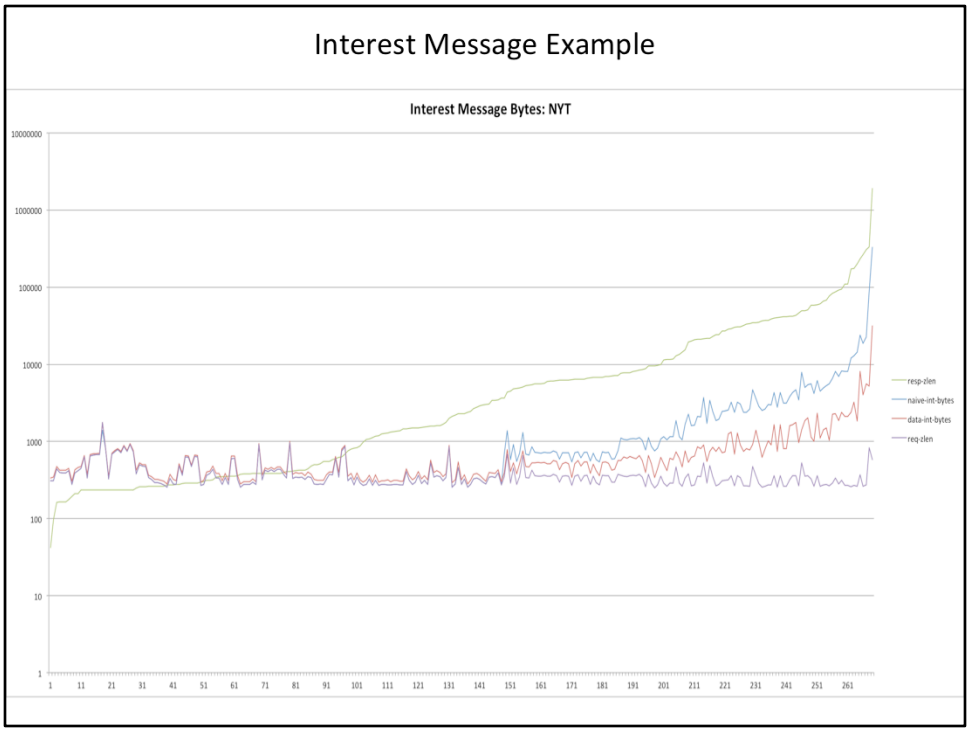
The blue bars are the 2-label name counts – we think these have some relation to the number of 'publishers' involved – content, ad networks, trackers, frameworks, etc. Again, we plot the average on a horizontal line.

Getting to the CDN via DNS

- Urls' names use DNS CNAME chains for indirection
- The DNS records (mostly) have very short ttls, giving the authoritative DNS servers the opportunity to generate updated records frequently

```
mjs-rmac:~ mjs$ dig graphics.nytimes.com A

; <<>> DiG 9.8.5-P1 <<>> graphics.nytimes.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 49901
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 8, ADDITIONAL: 8

;; QUESTION SECTION:
;graphics.nytimes.com.        IN    A

;; ANSWER SECTION:
graphics.nytimes.com.   60    IN    CNAME static.nytimes.com.edgesuite.net.
static.nytimes.com.edgesuite.net. 11168   IN CNAME a1158.g1.akamai.net.
a1158.g1.akamai.net.    20    IN    A     63.80.4.160
a1158.g1.akamai.net.    20    IN    A     63.80.4.202

[...]
```

This is an illustration from the nytimes site of one of the DNS-based mechanisms used to move requests to a well-known CDN network. We resolve a name that appears to be part of the Times itself, but we're led through a chain of CNAMEs to a pair of A RRs in an akamai zone. The browser will use those A RRs (probably just the first one) as the destination of a tcp connection. The short ttls on the A records compel the host's local stub resolver (and any intervening recursive resolvers) to refresh the A records frequently, giving akamai the opportunity to manage load. There's obviously some cost to the client here – the recursive resolver (usually at the client's business or ISP) is performing several queries, with their associated RTTs.



This shows a sample from the NYT home page. The objects are sorted by size - the pale green line is the object sizes. Note that this uses a log scale on the Y axis. The bottom line is the http request size (gzipped). A couple of the things we've discussed show up – there are quite a few requests that are larger than the returned data, and most of the objects are < 4KB, and it's just interesting to see that in a little detail. The largerst, right-most object is an embedded video, ~1.6MB.

The red and blue lines show a couple of simple NDN models applied to these data. The blue line shows a sort of 'naive' model, where the Interest names have to carry all of the request data. Since the names are large, once we get to the larger data objects we have to send more and more Interest messages to retrieve the same web object. The red line shows an alternative model, where we add an identifier to the base object name (the location, path, and arguments) and carry the headers in a separate TLV. The shorter name leaves more room in the returned Content messages, so we're able to retrieve objects with fewer Interests in this model, sending fewer request bytes than in the 'naive' model.

NDN at a Disadvantage?

- Carrying client-side cookie and header data in names will be costly
  - Or the entire stateless/RESTful model has to be replaced
- How effective can in-network caching be when the commercial web expects to hear from (and measure, and track) clients?
- How should the client stack manage the Interest/Content cycles for dozens or hundreds of objects?
- CDN load-balancing depends on some fairly lively connection between the server load and the 'route'.
  - Does anyone think that route updates are going to be injected and distributed at rates approaching the DNS ttls we see?
  - Does the mystical magical strategy layer solve this?
- Can client source/location anonymity survive commercial pressure?

NDN Advantages?

- Some rtt costs are eliminated: DNS lookups, tcp connections
  - For small objects, a single Interest/Content cycle may be possible (depending on the mechanism used to deliver headers)
  - But NDN will have to retrieve some keys
- Eliminating tcp changes congestion behavior
  - multiple connections with larger initial iw dump many packets on the wire before any (sender-side) congestion control is possible
  - in-network, multipath congestion control
- Elaborate mechanisms have evolved to move requests onto CDNs and across administrative domains
  - Can NDN offer some operational advantage somewhere?

Initial Site List

boston.craigslist.org
en.wikipedia.org
imgur.com
mail.google.com
www.amazon.com
www.boston.com
www.cnn.com
www.crateandbarrel.com
www.ebay.com
www.huffingtonpost.com
www.imdb.com
www.intellicast.com
www.nytimes.com
www.sears.com
www.tomshardware.com
www.yahoo.com