

# On the Effectiveness of DNS-Based Server Selection



Anees Shaikh, Renu Tewari



Mukesh Agrawal

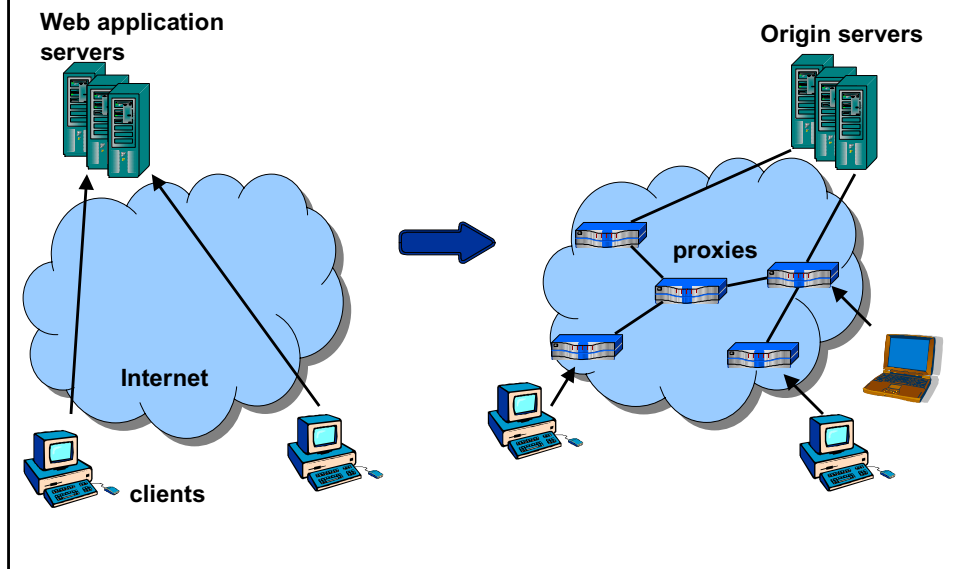
Carnegie Mellon

## Outline



- Context and motivation
- Why and how DNS server selection works
- Problems with DNS server selection
- Our focus and findings
- Concluding remarks

## Accessing Web services

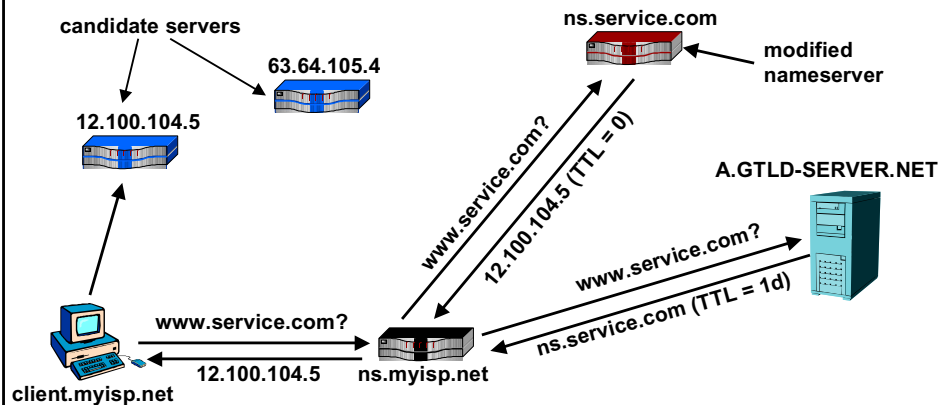


## Server selection

- Choose the “best” point of service
  - least loaded server
  - smallest round-trip delay to client
  - smallest number of hops to client (AS or network)
  - server with the requested content
- Techniques
  - Application layer (HTTP redirect)
  - Transport layer (TCP connection splicing/handoff)
  - **DNS**
    - popularized by CDSPs and equipment/software vendors

## DNS server selection

- simple – no change to existing protocols, clients, or servers
- general – works for any IP-based application



## Problems with DNS approach

- Small TTLs to improve load-balancing
  - reduce caching and increase client latency
  - increase load on the DNS
- Local nameserver  $\neq$  client
  - more difficult to pick proximal server
- *Our focus:*
  - What is the impact of disabling caching on client-perceived latency?
  - How proximal are clients and their local nameservers?

## Impact of small TTL values

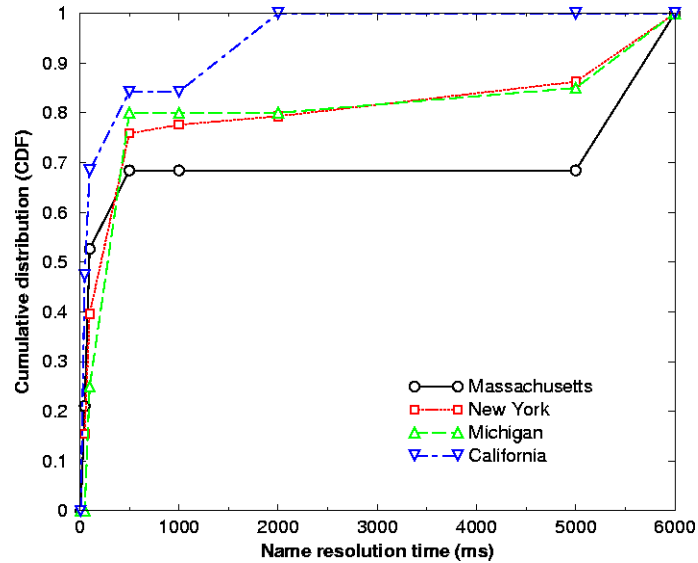
- Overhead of single name resolution
  - measure resolution time with varying amount of caching
  - compare against page download time
- Multiple name resolutions for a single page
  - determine embedded object distribution in typical Web pages
  - compute name resolution overhead for downloading multiple objects with caching disabled

## Measuring DNS latency

- Measure latency from four sites using hostnames from ISP log and popular sites
- Consider three caching scenarios
  - no authoritative name server, no server address
  - cached auth. nameserver, no server address
  - cached server address
- BIND 8.2.1, no client application caching

Cache contents	Median latency (NY)
root and .com only	200 ms
auth. nameserver	60 ms
server address	2.3 ms

## Name resolution latency



## Impact of embedded objects



- characterize object distribution

- measure page and per object download time

- DNS lookup overhead (min. caching)

- one lookup  $\frac{200 \text{ ms}}{6 \text{ sec}} = 3\%$

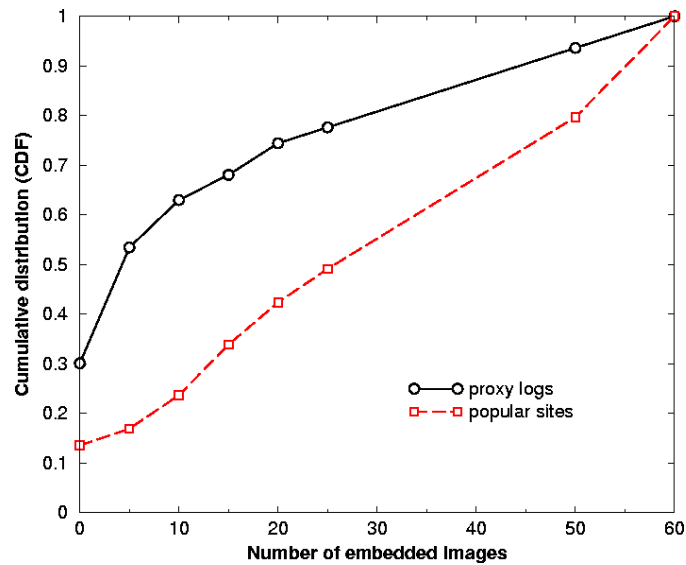
- every object  $\frac{200 \text{ ms}}{415 \text{ ms}} = 48\%$

Today's popular pages:

- 3.7 lookups per page

- 1 name lookup for every 8 objects

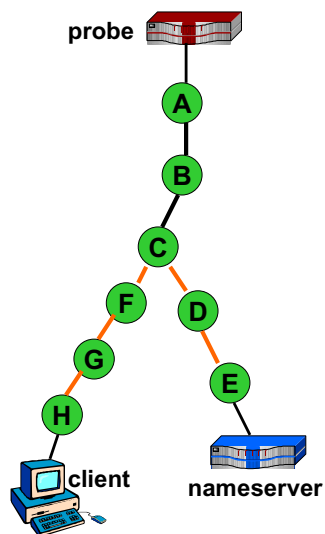
## Embedded object distribution



## Client-nameserver proximity

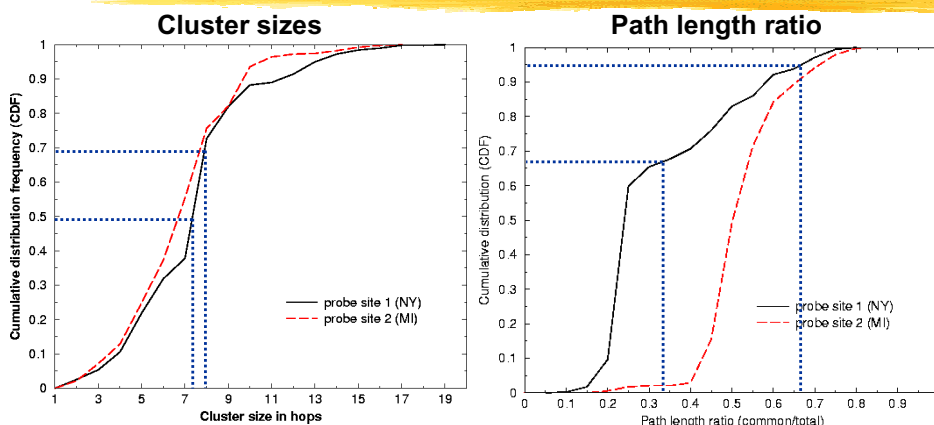
- Measure “similarity” of network paths to some point in the network (e.g., a server)
- Methodology
  - collect and analyze client-nameserver pairs
  - combined DNS and HTTP logs
  - dial-up ISP accounts
    - 11 U.S. ISPs, ~50 POPs per ISP (avg.)
    - 1090 dial-up client-nameserver pairs

## Client-nameserver clusters



- tracert to client and ns from 2 probe sites (MI and NY)
- cluster = max disjoint path length
- compute ratio of common/total path length

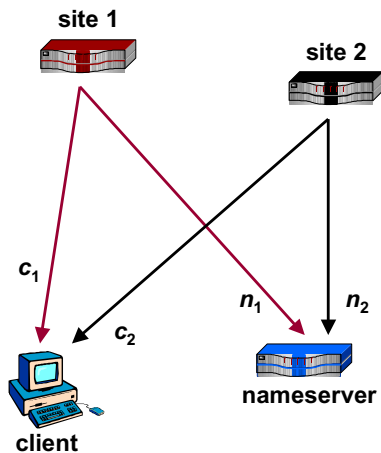
## Client-nameserver paths



- 30% of clusters are  $\geq 8$  hops
- median cluster size is  $\sim 7$  hops

- disjoint path at least twice as long for 65% pairs
- common path at least twice as long for 5% pairs

## Client vs. nameserver latency



- Are measurements to NS representative of clients?
  - | If nameserver is closer to site 1 than site 2, same for client?
  - | Yes, in 79% of cases tested
- Correlation of NS latency to client latency
  - |  $(n_1 - n_2)$  vs.  $(c_1 - c_2)$
  - | positive, very weak correlation ( $\rho = 0.32$ )

## Summary

- Small TTL values
  - | reduced caching can grow response time by up to two orders of magnitude
  - | increasing off-server embedded objects incurs significant DNS overhead (worst case: 50%)
- Client-nameserver proximity
  - | clients and nameservers often have highly disjoint paths to servers
  - | latency to NS is a weak predictor of latency to actual client

## Remarks



- TTL values are tricky
  - balance responsiveness against increased client latency and decreased scalability
  - use large TTL values and rely on DNS only for coarse load balancing
- Amortize lookups and co-locate objects
- Better mechanisms for proximity-based server selection
  - modifications to DNS protocol