Accelerating Last-Mile Web Performance with Popularity-Based Prefetching

Srikanth Sundaresan, Nazanin Magharei, Nick Feamster, Renata Teixeira
{srikanth.sundaresan,nazanin,feamster}@cc.gatech.edu, renata.teixeira@lip6.fr

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network Management*; C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network Operations*

General Terms

Measurement, Performance

Keywords

Broadband Networks, Web performance, Pre-fetching

1. INTRODUCTION

As broadband speeds increase, latency is becoming a bottleneck for many applications—especially for Web downloads. Latency affects many aspects of Web page load time, from DNS lookups to the time to complete a three-way TCP handshake; it also contributes to the time it takes to transfer the Web objects for a page. Previous work has shown that much of this latency can occur in the last mile [2]. Although some performance bottlenecks can be mitigated by increasing downstream throughput (*e.g.*, by purchasing a higher service plan), in many cases, latency introduces performance bottlenecks, particularly for connections with higher throughput.

To mitigate latency bottlenecks in the last mile, we have implemented a system that performs DNS prefetching and TCP connection caching to the Web sites that devices inside a home visit most frequently, a technique we call *popularity-based prefetching*. Many devices and applications already perform DNS prefetching and maintain persistent TCP connections, but most prefetching is predictive based on the content of the page, rather than on past site popularity. We evaluate the optimizations using a simulator that we drive from traffic traces that we collected from five homes in the BISmark testbed [1]. We find that performing DNS prefetching and TCP connection caching for the twenty most popular sites inside the home can double DNS and connection cache hit rates.

2. POPULARITY-BASED PREFETCHING

Figure 1 shows the design of our system, which we have implemented and deployed on BISmark. We augment `dnsmasq`, a caching DNS resolver, to maintain a *popular domain list* that users in the home network resolve most frequently, and `polipo`, a HTTP proxy, to maintain a *popular connection list* of domains users visit. A helper script refreshes the DNS entries for expired domains, and maintains open connections to servers in the connection list by periodically sending dummy `GET` requests to each site. (The prototype ensures that neither of these actions induce a positive feedback loop that keeps unpopular domains in the popular domains list.) The following parameters affect the tradeoff between page load time and overhead:

- *Sizes of the popular domains and popular connections list.* The number of domains for which the system actively prefetches DNS records and the number of sites to which it maintains active connections. The router maintains this list using an (LRU) cache.
- *Domain and connection timeout thresholds.* If no device in the home network looks up a domain in the popular domain list within a *domain timeout threshold*, the system removes the domain from the list and no longer actively prefetches the DNS records for this domain. Similarly, if no HTTP request occurs within a *connection timeout threshold*, the system no longer keeps TCP connections alive for that domain. The system maintains only one active connection per domain.

Deploying these optimizations on the router itself, rather than relying solely on browser-based optimizations, offers several benefits, since: (1) not all devices and browsers may implement these optimizations; (2) users and devices across a household may have common browsing activity.

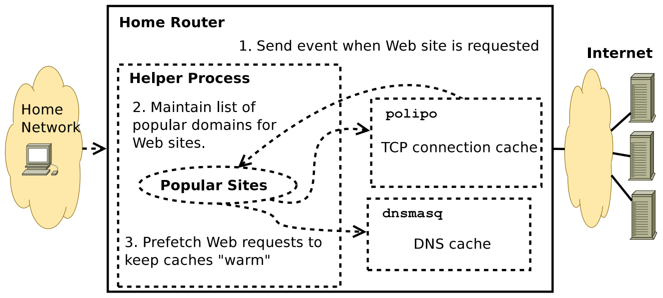
Home Router
1. Send event when Web site is requested
Internet
Helper Process
Home Network
2. Maintain list of popular domains for Web sites.
polipo
TCP connection cache
Popular Sites
dnsmasq
DNS cache
3. Prefetch Web requests to keep caches "warm"

Figure 1: *Augmenting the home router to automatically prefetch DNS records and maintain TCP connections to popular Web sites.*

3. EVALUATION

We analyze the effects of popularity based DNS prefetching and connection caching using a trace-driven discrete event simulator that models the optimizations as described in the previous section.

3.1 Simulator Setup

We evaluate the benefits of popularity-based prefetching using a simulator driven by traces from five homes in the BISmark testbed [1]. These traces contain all DNS lookups and HTTP connection requests (anonymized) across all devices. Table 1 shows

	Home 1	Home 2	Home 3	Home 4	Home 5
Statistics					
People	1	3	1	2	1
Days	81	75	88	103	24
DNS Lookups	159K	185K	82K	219K	60K
New HTTP transactions	450K	254K	146K	347K	99K
Traffic Properties					
Unique Lookups	23	5	38	9	20
Unique HTTPs	32	8	23	10	14
DNS Int-req.	840s	450s	850s	180s	502s
Non-simultaneous HTTP intervals	210s	63s	135s	72s	340s
Non-simultaneous HTTP requests	59%	58%	37%	60%	63%

Table 1: *Properties of traces we have analyzed in our simulations.*

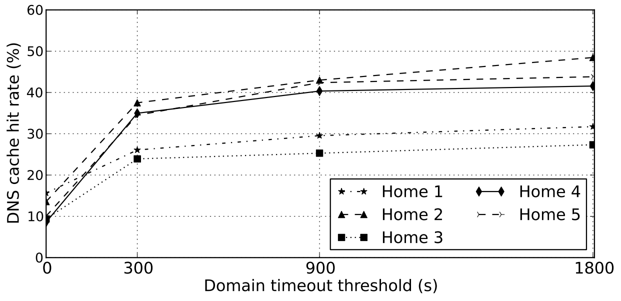

Figure 2: *Daily DNS cache hit ratio with DNS prefetching when the popular domain list size is 20.*

the characteristics of the homes we study. Unique lookups and HTTPs are the median number of unique domains and new HTTP connections every hour. DNS inter-request time is the median inter-arrival time between DNS requests. Non-simultaneous HTTP interval is the median time between the end of one HTTP flow and the start of the next to the same server, for non-overlapping flows. Non-simultaneous HTTP requests is the percentage of HTTP flows that do not overlap with the previous request to the same server.

DNS caching and prefetching The simulator maintains the DNS cache using the timestamp, domain, and TTL of resolved DNS responses from traffic logs. When prefetching a domain, the simulator introduces a random delay between 50 and 200 ms; the trace determines the TTL that we use for each DNS record. We consider DNS requests for domains that wait for a response as cache misses.

TCP connection caching The simulator maintains the TCP connection cache based on the timestamp, five-tuple, and duration of both the entire TCP connection and its connection establishment of all port 80 requests from the passive traces. For each new request, we update the popularity of the corresponding IP address and compare it with the entries in connection cache. In the event of a cache miss, or when a connection to the same IP address is still in use by another request, we establish a new connection, with the flow and TCP connection establishment durations that correspond to the TCP connection in the trace. Upon a cache hit, the simulator denotes the cache entry as in use for the duration of TCP connection, minus the connection establishment time.

3.2 DNS prefetching

The size of the popular domains list and the value of the domain timeout threshold determine the benefits and the overhead of DNS prefetching. The overhead is the ratio of the number of prefetched DNS requests to the total number of DNS cache hits. First, we study how the domain timeout threshold affects the DNS cache hit ratio and overhead when the size of popular domain list is 20.

Figure 2 shows the results of our experiment. The median daily DNS cache hit ratio without prefetching varies from 7% to 15% depending on the home, but setting a prefetching threshold of 30 minutes results in a cache hit ratio of about 30–50%. By default, the prefetching overhead varies from 5 to 150, but applying the backoff algorithm described in Section 2 reduced the prefetching overhead to less than 10 for all homes, with less than a 2% reduction in the hit ratio. The DNS cache hit ratio and prefetching overhead depend on traffic characteristics in each home, such as number of concurrent Internet users, diversity of the looked up domains, inter-arrival times of of the requests, browser, and distribution of TTLs across domain names. Larger DNS request interarrival times decrease the DNS cache hit ratio. Homes with higher prefetching overhead have domains with TTL values as low as one second. To test the sensitivity of our results to the domain list size, we repeated the experiment for sizes ranging from 10 to 320. Sizes beyond 20 results in only a marginal improvement in cache hit ratio in most cases.

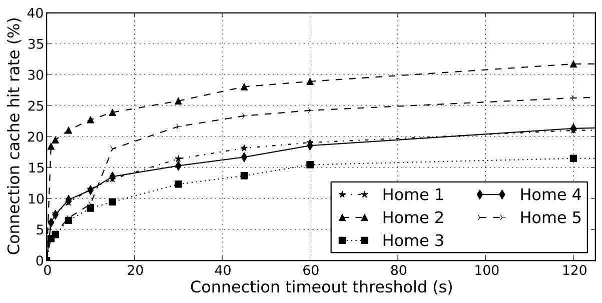

Figure 3: *Daily TCP connection cache hit ratio when the popular connection list size is 20.*

3.3 TCP connection caching

We explore whether popularity-based TCP connection caching can improve the cache hit ratio. We use a popular connection list size of 20. We define overhead as the number of keepalive packets required to keep the TCP connections to the server open.

We find that maintaining an active connection cache of popular connections can improve connection cache hit ratio of as much as 30%, even by maintaining no more than one active connection to each server. Figure 3 shows that hit ratio of the connection cache can improve by up to 30%, even for connection timeout values of as small as two minutes. For a timeout of two minutes, the connection hit ratios are 16–32%, compared with a baseline of 4–19% across homes. The overhead varies from two to nine keep-alives per cached connection per day with a timeout value of two minutes. Increasing the timeout to larger than one minute yields only marginally better connection hit ratio. Even small threshold values result in significant improvements because HTTP requests tend to be quite bursty. Due to the low diversity in HTTP requests, we observe only a 2% improvement in daily connection cache hit ratio for popular connection list size of larger than 20.

Acknowledgments This project is supported by NSF CNS-1059350 and a Google Focused Research grant.

REFERENCES

[1] BISMark: Broadband Internet Service Benchmark. `http://projectbismark.net/`.

[2] S. Sundaresan, W. de Donato, N. Feamster, R. Teixeira, S. Crawford, and A. Pescapè. Broadband internet performance: A view from the gateway. In *Proc. ACM SIGCOMM*, Toronto, Canada, Aug. 2011.