

What do you call a proxy cache that does not make the Web faster?

?

What do you call a proxy cache that does not make the Web faster?

A success!

(as long as it doesn't make it slower, either)

# Why build a proxy cache that doesn't improve Web response times?

---




# Why build a proxy cache that doesn't improve Web response times?

---

- ◆ You like to keep your machine room warm.

# Why build a proxy cache that doesn't improve Web response times?

---

- ◆ You like to keep your machine room warm. 
- ◆ You're in a down-sized business and are trying to keep all the extra machines busy.

# Why build a proxy cache that doesn't improve Web response times?

---

- ◆ You like to keep your machine room warm. ❌
- ◆ You're in a down-sized business and are trying to keep all the extra machines busy. ❌
- ◆ You want to save money by using less bandwidth, but don't want to improve service to your users.

# Why build a proxy cache that doesn't improve Web response times?

---

- ◆ You like to keep your machine room warm. ❌
- ◆ You're in a down-sized business and are trying to keep all the extra machines busy. ❌
- ◆ You want to save money by using less bandwidth, but don't want to improve service to your users. ❌
- ◆ You want to evaluate a new caching policy without influencing user behavior.

# Why build a proxy cache that doesn't improve Web response times?

- ◆ You like to keep your machine room warm. ❌
- ◆ You're in a down-sized business and are trying to keep all the extra machines busy. ❌
- ◆ You want to save money by using less bandwidth, but don't want to improve service to your users. ❌
- ◆ You want to evaluate a new caching policy without influencing user behavior. ❌
- ◆ You are implementing the SPE architecture. ✓

# When does a hit = a miss?

---

Brian D. Davison, CSE, Lehigh University  
Chandrasekar Krishnan, Fatline Corporation  
Baoning Wu, CSE, Lehigh University

Sponsored by NSF grant ANI 9903052.

# Talk Outline

---

- ◆ Motivation and introduction
- ◆ Background
  - ◆ Web proxy evaluation concerns
  - ◆ The simultaneous proxy evaluation architecture
- ◆ Proxy implementation
- ◆ Implementation issues
- ◆ Evaluation of implementation
- ◆ Summary and future directions

# Web Proxy Evaluation

- ◆ Want to be able to compare performance.
  - ◆ Many aspects: byte hit rates, response times, requests/sec.
- ◆ Evaluation choices:
  - ◆ Install proxy temporarily in real-world environment.
    - ◆ Dangerous! Potentially expensive.
  - ◆ Replay captured trace through proxy to real world content.
    - ◆ Content gets old quickly; need resources to replay accurately; may request content with side effects.
  - ◆ Place proxy within benchmark workload generator and content server.
    - ◆ Isolated network and artificial content may not reflect real world.
    - ◆ Content-based prefetching may fail to be useful.
    - ◆ e.g., Wisconsin Proxy Benchmark, SPECweb99, Web Polygraph

# Web Polygraph

NLANR/The Measurement Factory

<http://www.web-polygraph.org/>

- ◆ Benchmark for implemented proxy caches.
  - ◆ Free for unpublished comparisons.
- ◆ Periodic cache-off competitions by TMF.
- ◆ Uses artificial traces (with artificial content).
- ◆ Verifies some level of performance.
  - ◆ Measures page and byte hit rates, reqs/sec, etc.
- ◆ Reproducible.
- ◆ May not reflect specific real-world environments.

# Our complementary approach

---

## SPE (Simultaneous Proxy Evaluation) Architecture

*An evaluation architecture that simultaneously tests multiple proxies on a live user load and network connection.*

# Design Choices and Ramifications

---

- ◆ Choice: Use Live Request Loads
  - ◆ Artificial loads reflect selected aspects of real-world loads.
  - ◆ Captured logs can be used, but are often out of date.
- ◆ Choice: Use Full Content Via Real Network
  - ◆ Full content may be needed by some prefetching systems.
  - ◆ Need real headers (for accurate cacheability calculations).
  - ◆ Get real-world variations in networks and servers.
- ◆ Ramification: Give Up Reproducibility
  - ◆ Request loads, content, and networks change over time.
  - ◆ Can still have comparability (simultaneous tests).
  - ◆ Test on real-world environment.

# Simultaneous Proxy Evaluation

- ◆ Architecture features [Davison, WCW 1999]
  - ◆ Replicate real-world environment
    - ◆ Clients and servers see a single entity (a proxy)
    - ◆ Tested proxies see just a client and a parent proxy
  - ◆ Evaluate black-box proxies (including prefetching)
  - ◆ Measure performance and log results
    - ◆ Response times
    - ◆ Bandwidth used
  - ◆ Don't need to find failure modes

# SPE Architecture

Clients



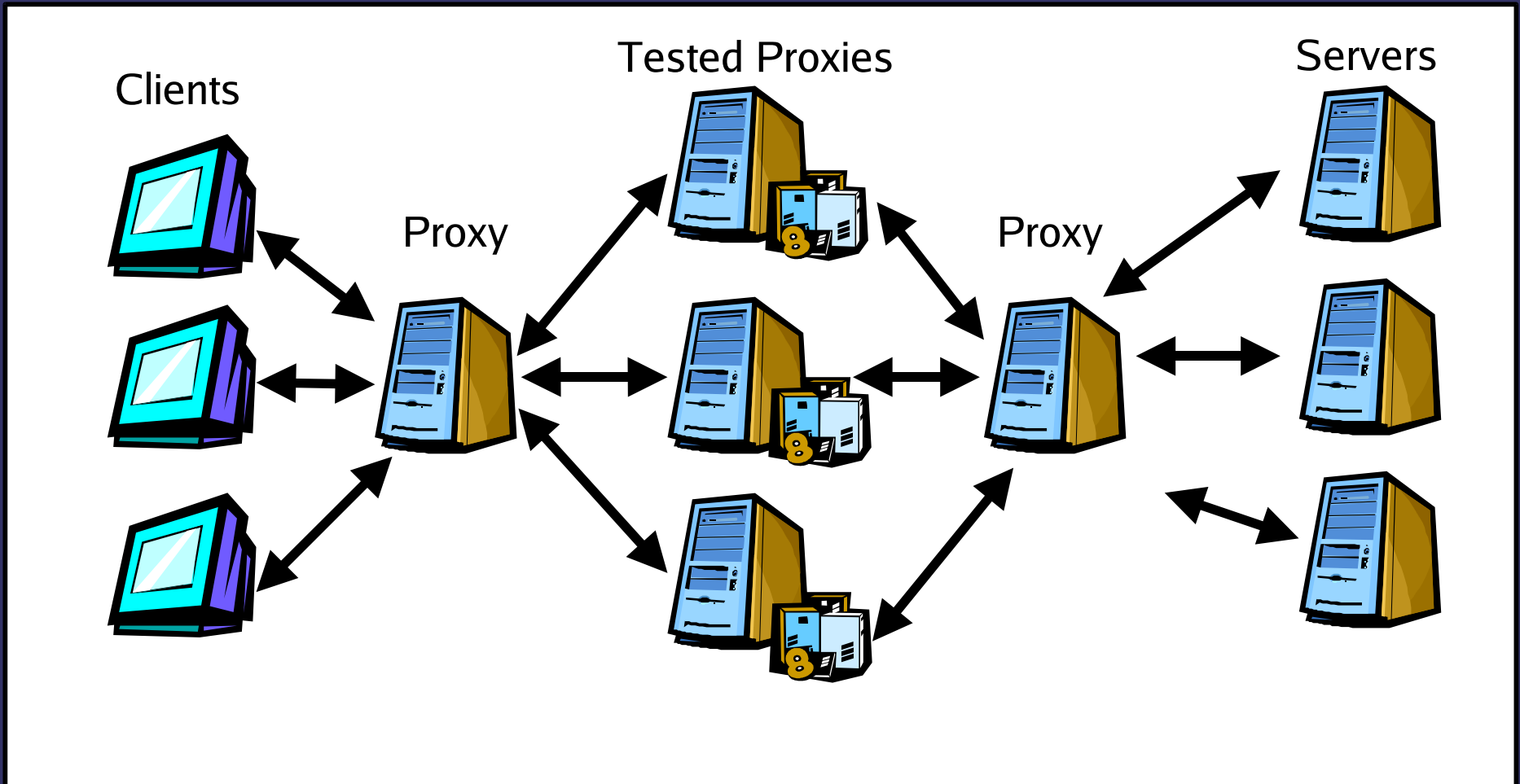
Tested Proxies



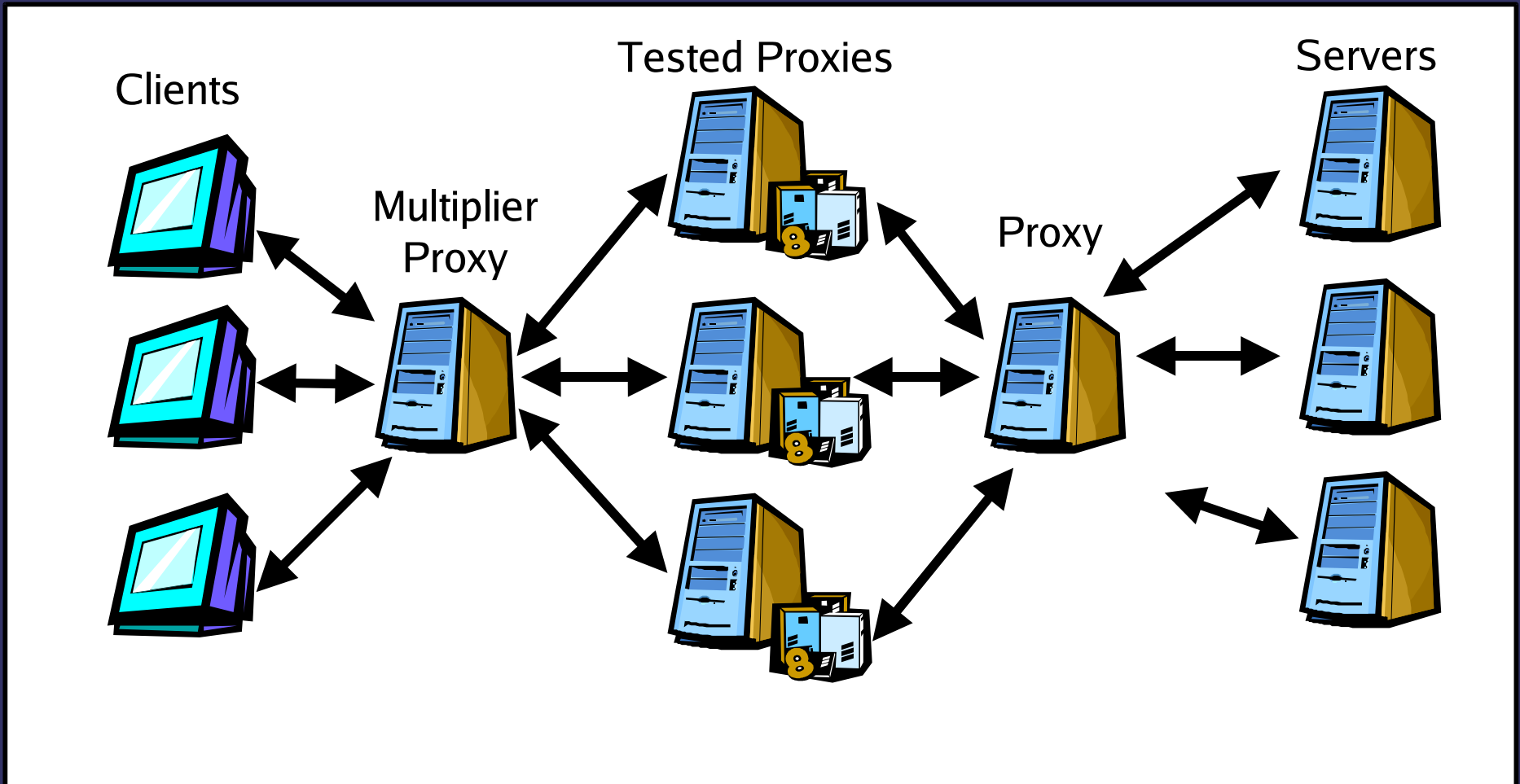
Servers



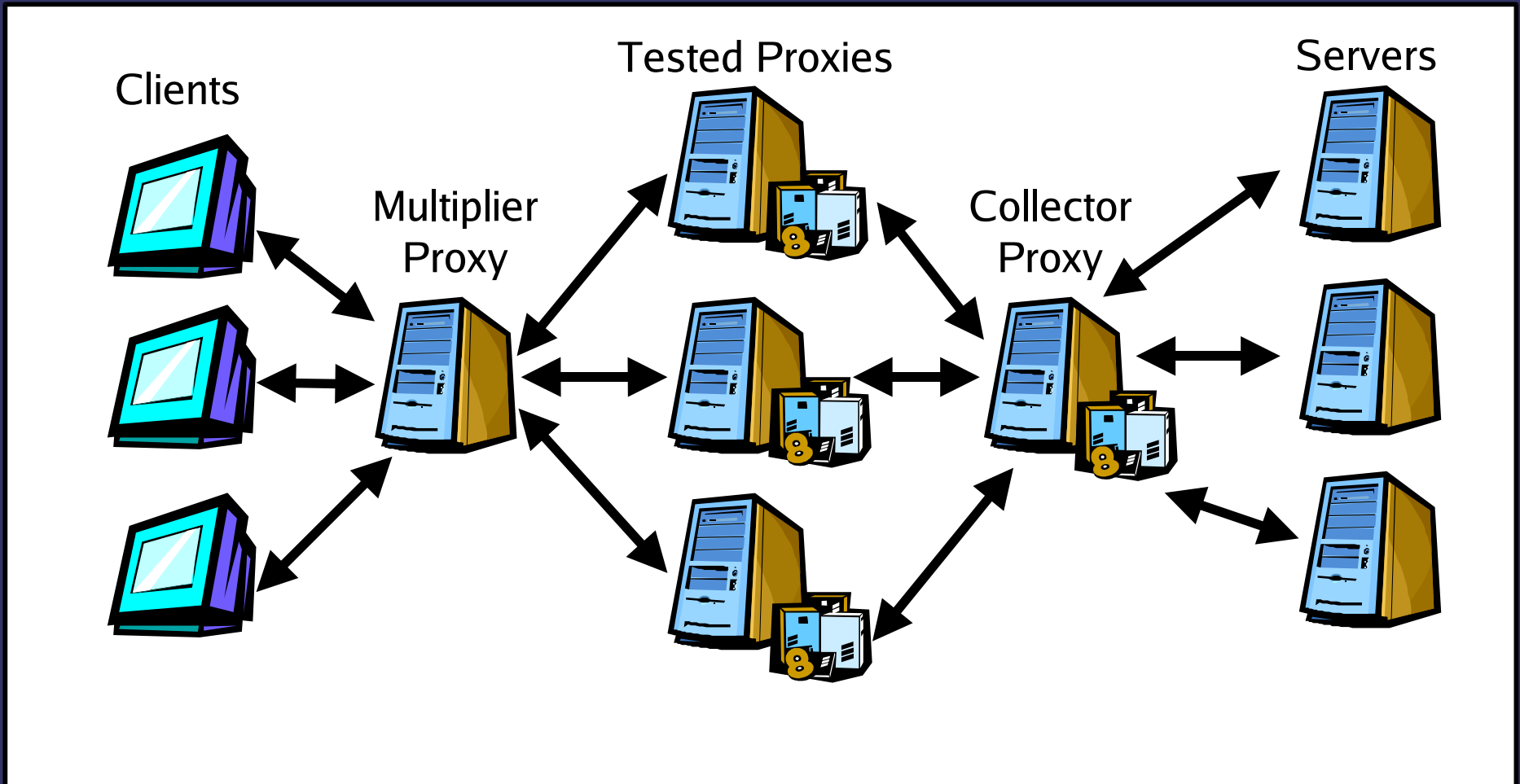
# SPE Architecture



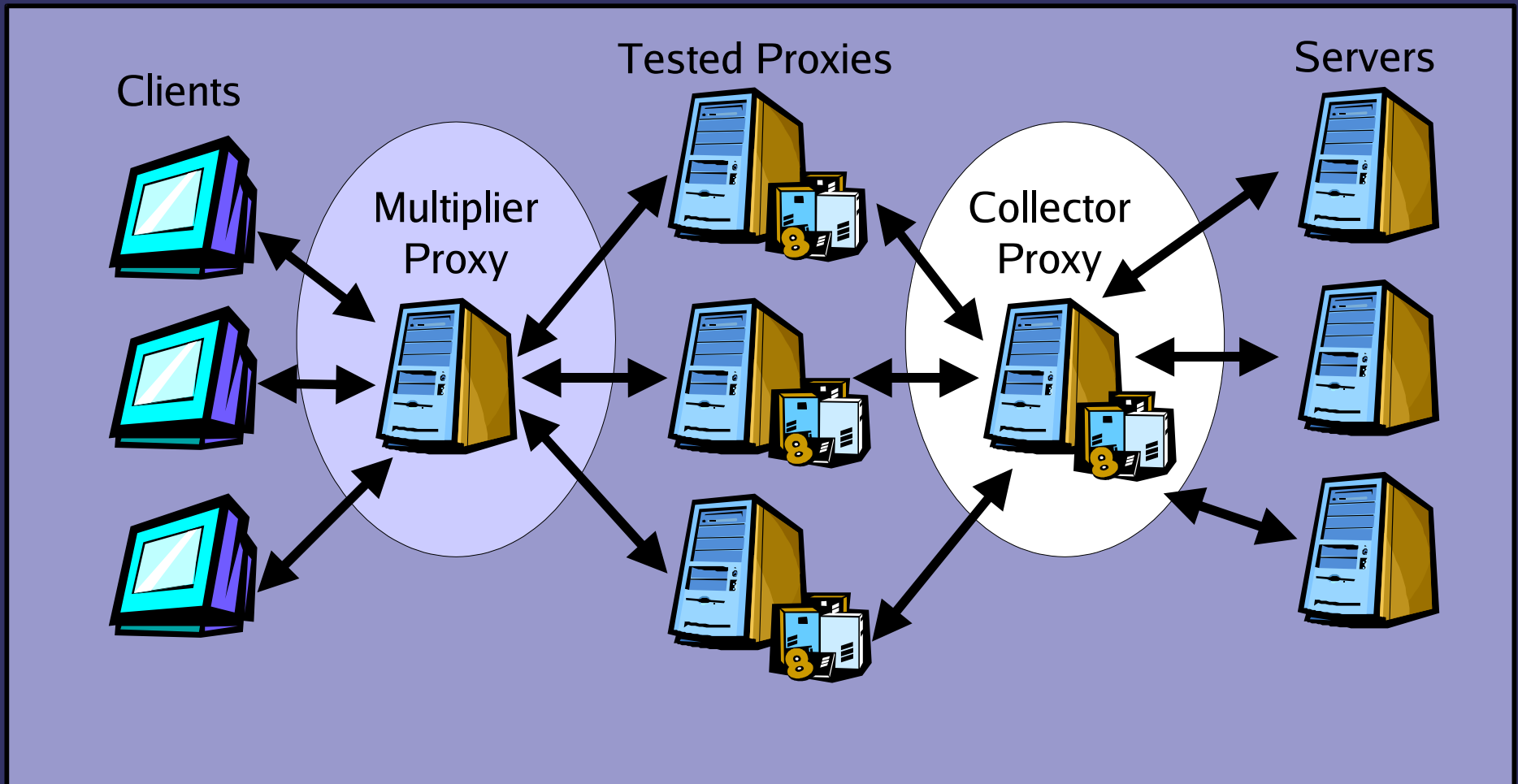
# SPE Architecture



# SPE Architecture



# SPE Architecture



Today's focus: a proxy that returns hits as slowly as a miss.

# Proxy Implementation

- ◆ Modified open-source Squid proxy cache.
  - ◆ No OS modifications.
- ◆ On a cache miss, we record
  - ◆ connection establishment time,
  - ◆ initial latency (first byte), and
  - ◆ transfer time (remaining bytes).
- ◆ On a cache hit, we insert delays to replicate the cache miss timings.

# Implementation Issues

---

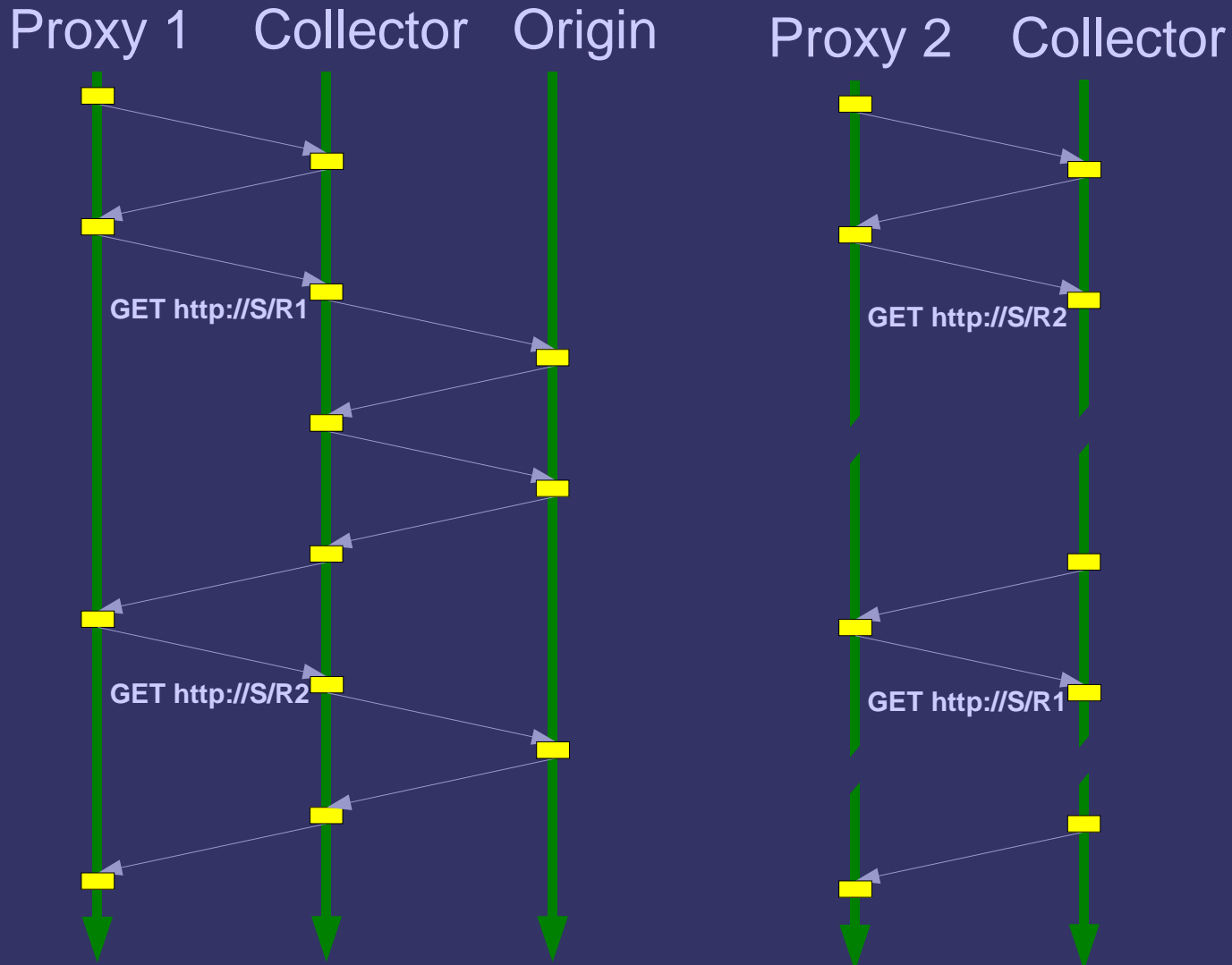
- ◆ Network stability
  - ◆ Measurements of access costs may be stale.
  - ◆ We chose consistent performance reproduction over accurate performance reproduction (which would preclude caching entirely).
  - ◆ We can limit caching periods to minimize adverse effects.

# Implementation Issues

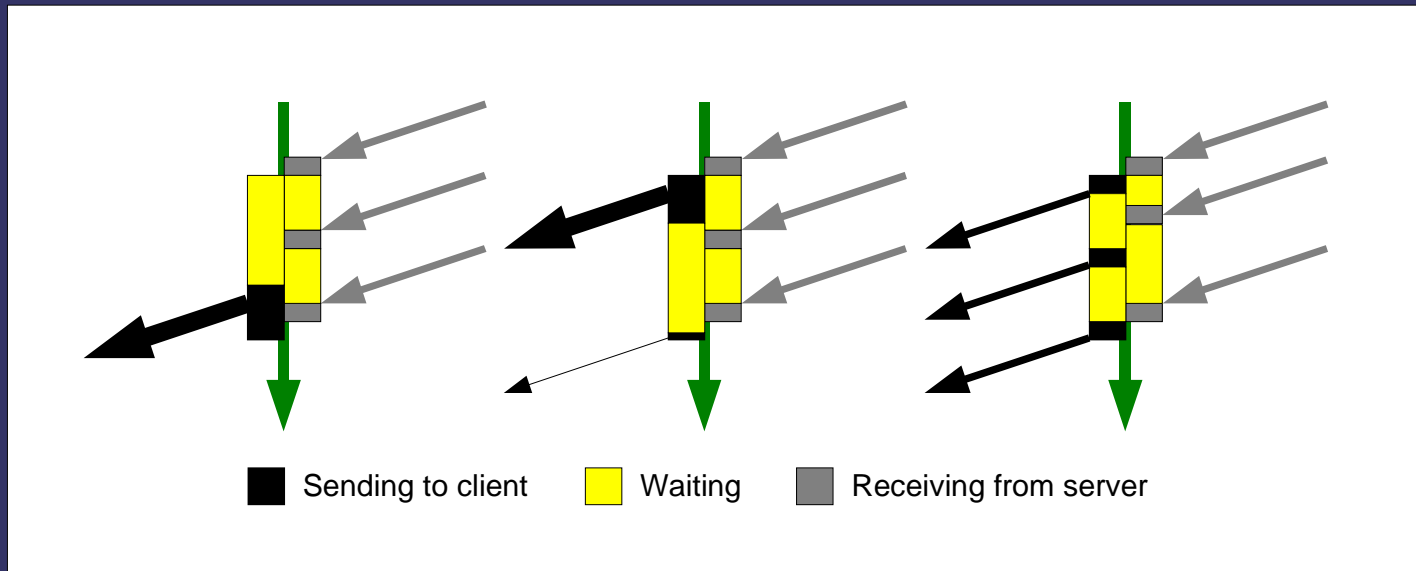
---

- ◆ TCP connection requests
  - ◆ Served at full speed on LAN.
  - ◆ Delay first response by the original connection time.
- ◆ Persistent connections
  - ◆ We do not use persistent connections to origin servers.

# Persistent Connections



# Inter-chunk Delay Schedules



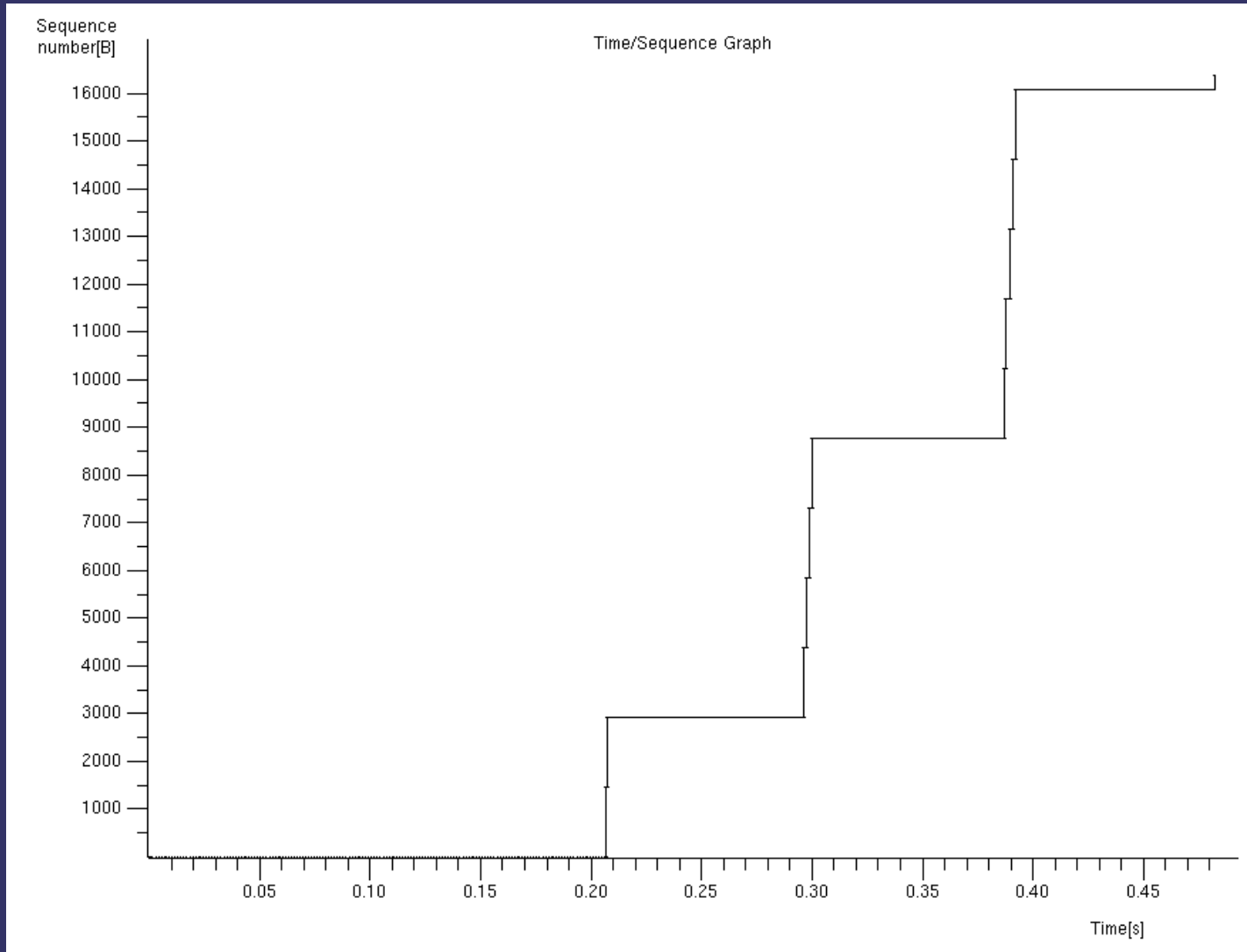
- ◆ Response rate from Collector
  - ◆ Ave. of the origin server (not served in original bursts)
- ◆ Schedule important because browser will request embedded resources whenever chunk contains links.

# Chunk Scheduling Experiment

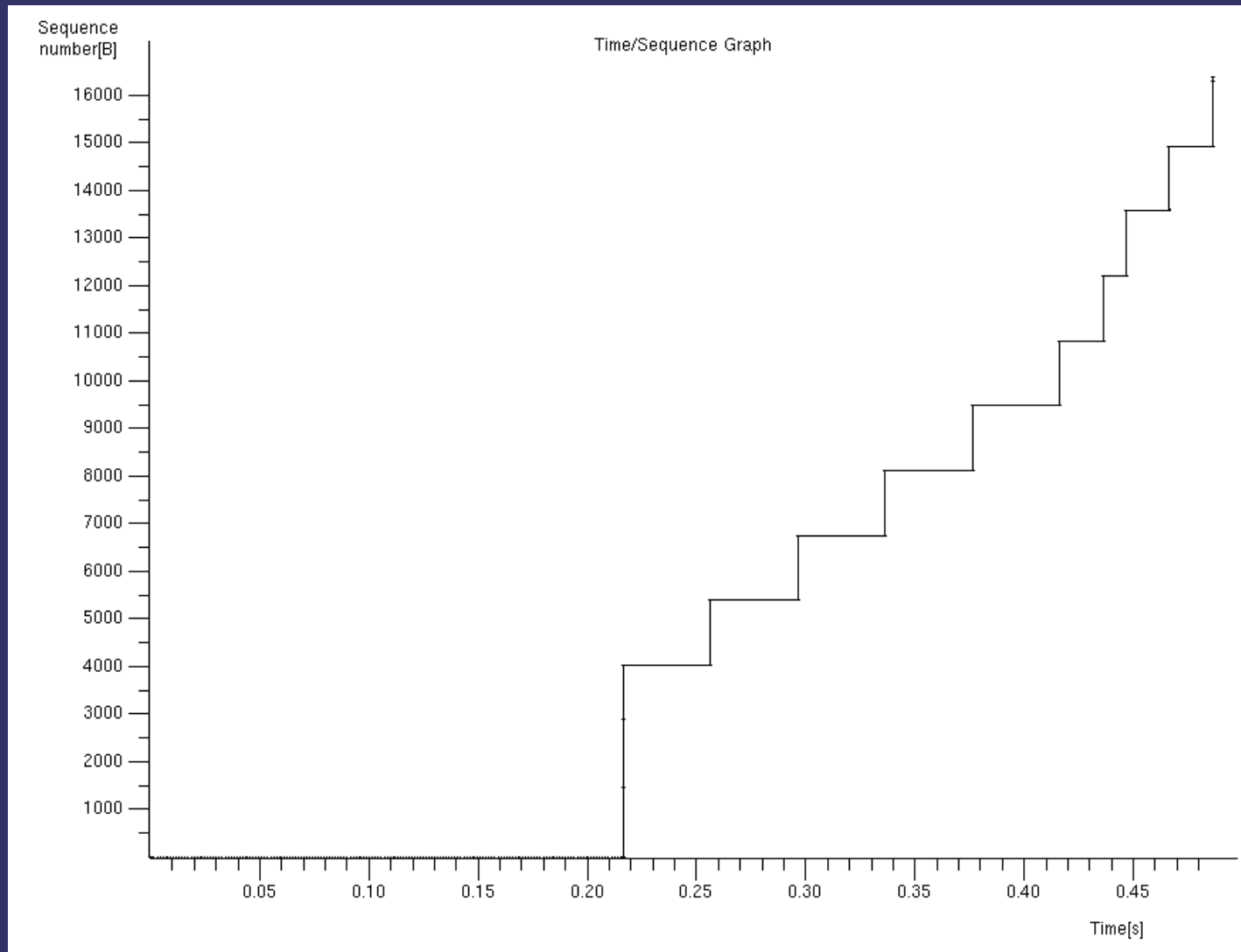
---

- ◆ Demonstrate chunk scheduling in our implementation for a single request.
  - ◆ Show byte arrival times for cache miss.
  - ◆ Show byte arrival times for hit with replicated delays.
- ◆ Difficult to replicate at application layer because OS scheduling granularity is ~10ms.

# Original Time-Sequence



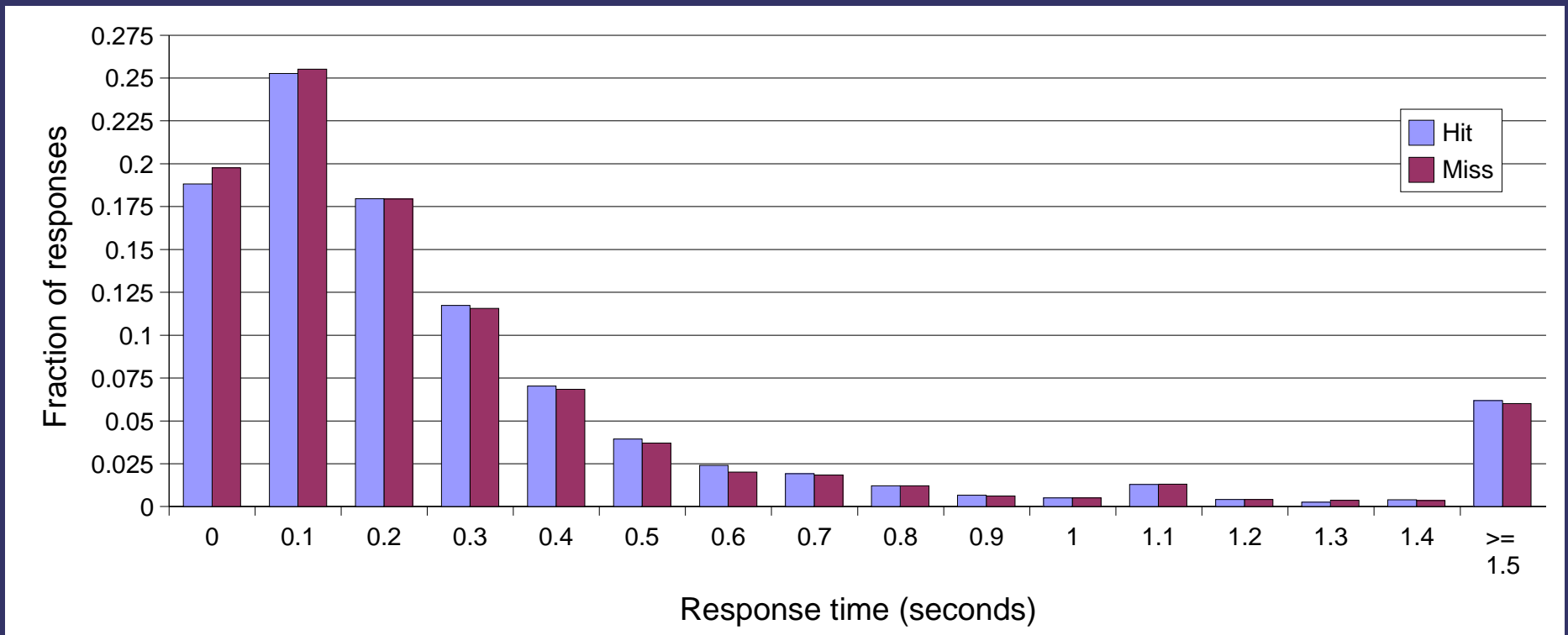
# Replicated Time-Sequence



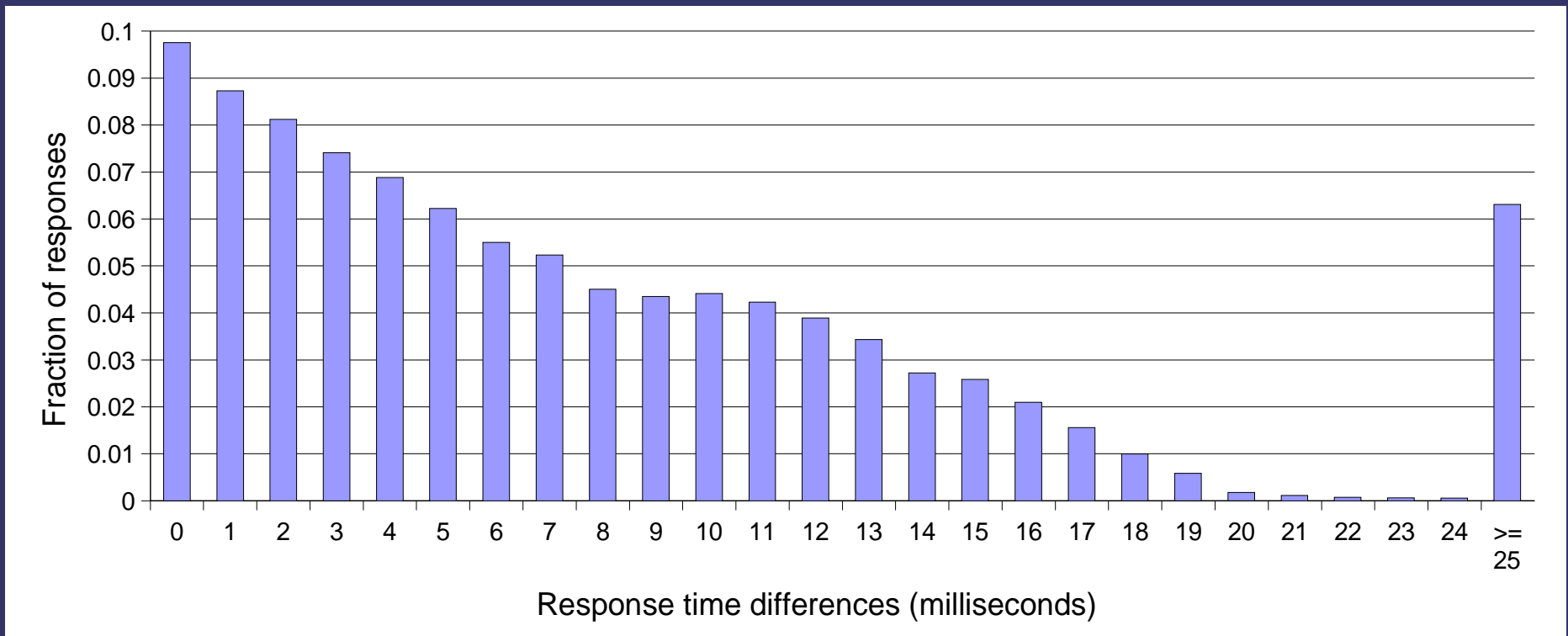
# Implementation Test

- ◆ Demonstrate ability to replicate the cost of a miss, even though the response is served from cache.
- ◆ Used httpperf with artificial trace -- subset of real URLs (from IRCache logs)
  - ◆ First request is a miss, Collector fetches from origin
  - ◆ Second request is served by the Collector from cache
- ◆ Present distribution of response times and differences between response times

# Hit & Miss Response Time Distrib.



# Miss/Hit Response-Time Differences



Median: 6.34, Mean: 7.44ms when all responses are cached.

Compare to Miss/Miss median of 10.31 and mean of 337.84ms.  
and unmodified Miss/Hit median of 245.62ms and mean of 503.34ms.

# Summary

- ◆ Ridiculed the idea of a proxy cache that did not decrease client response times.
- ◆ Reviewed the motivation and design of the simultaneous proxy evaluation architecture.
- ◆ Implemented one aspect needed for a SPE prototype:
  - ◆ A proxy cache that serves hits as slowly as misses.
- ◆ Discussed potential problems and our solutions.
- ◆ Verified the performance of our implementation.

# Future Directions

---

- ◆ Simultaneous Proxy Evaluation
  - ◆ Complete prototype with
    - ◆ Multi-threaded Multiplier
    - ◆ Collector that caches everything
  - ◆ Sample live real-world requests
  - ◆ Use prototype to evaluate multiple proxies

# Additional Resources

- ◆ <http://www.cse.lehigh.edu/~brian/>
  - ◆ Brian D. Davison
  - ◆ [davison@cse.lehigh.edu](mailto:davison@cse.lehigh.edu)
- ◆ <http://wume.cse.lehigh.edu/>
  - ◆ Papers and source codes
- ◆ <http://www.web-caching.com/>
  - ◆ Bibliography, conferences, products, caching and CDN news, other sites

