

Taming the WWW

From Web Measurements to Web Protocol Design

Azer Bestavros

Computer Science Department

BOSTON UNIVERSITY

Thursday July 25th, 1996

Software and Systems Research Center, Bell Labs, Murray Hill

July 26, 1996

Prefetching Techniques for the Web

Azer Bestavros

Computer Science Department

BOSTON UNIVERSITY

Friday July 26th, 1996

Overview of OCEANS Projects

— **Client-based Studies**

- ◇ (Client Caching Protocols)
- ◇ Prefetching Protocols
- ◇ (Dynamic Server Selection Protocols)

— **Server-based Studies**

- ◇ (Server Caching Protocols)
- ◇ Dissemination Protocols
- ◇ Speculative Service Protocols

— **Performance Modeling Studies**

- ◇ Web Traffic Characteristics
- ◇ (Web Access Patterns Characteristics)

Locality of Reference in a Client-Server Environment

Locality of Reference Flavors

◇ **Temporal:**

A document accessed frequently in the past is likely to be accessed again in the future.

◇ **Spatial:**

A document “neighboring” a recently accessed document is likely to be accessed in the future.

◇ **Geographical:**

A document accessed by a client is likely to be accessed in the future by “neighboring” clients.

Locality of Reference in a Client-Server Environment

How to Capitalize on it?

- ◇ On the client side, use “caching” and “prefetching” (e.g. Distributed file systems, Sun NFS, AFS, [Standberg 1985, Morriss 1986, Howard 1988], Proxy caching [Danzig 1993, Acharya 1993, Papadimitriou 1994], Cooperative client caching [Blaze 1993, Dahlin 1994]).
- ◇ On the server side, use “information dissemination” [Bestavros 1994], “geographical caching” [Braunh and Claffyh 1994], “speculative service” [Bestavros 1995], “geographical push caching” [Gwertzman and Seltzer 1995].

Motive

- ◇ The scalability of Internet services hinges on efficient distribution and partitioning of system resources to reduce the amount of data that must be moved.

Information Caching versus Information Dissemination

Information Caching

- ◇ Initiated by a client or a group of clients.
- ◇ Geared towards reducing service time.
- ◇ Relies on temporal locality of client reference patterns.
- ◇ Ensuring consistency is expensive.

Information Dissemination

- ◇ Initiated by servers.
- ◇ Geared towards balancing load and reducing traffic.
- ◇ Relies on temporal/geographical popularity of documents.
- ◇ Ensuring consistency is cheap.
- ◇ Requires collaboration of “server proxies”.

Client-initiated Caching Study

Experiment Description

- ◇ We instrumented Mosaic to log all user accesses on our site [BCC:95].
- ◇ We studied cache performance at various levels:
 - Session Caching: One cache per session
 - Host Caching: One cache per host
 - LAN Caching: One cache per LAN
- ◇ We used the logs obtained from Mosaic to perform trace simulations for various protocols [BCCCHM:95].

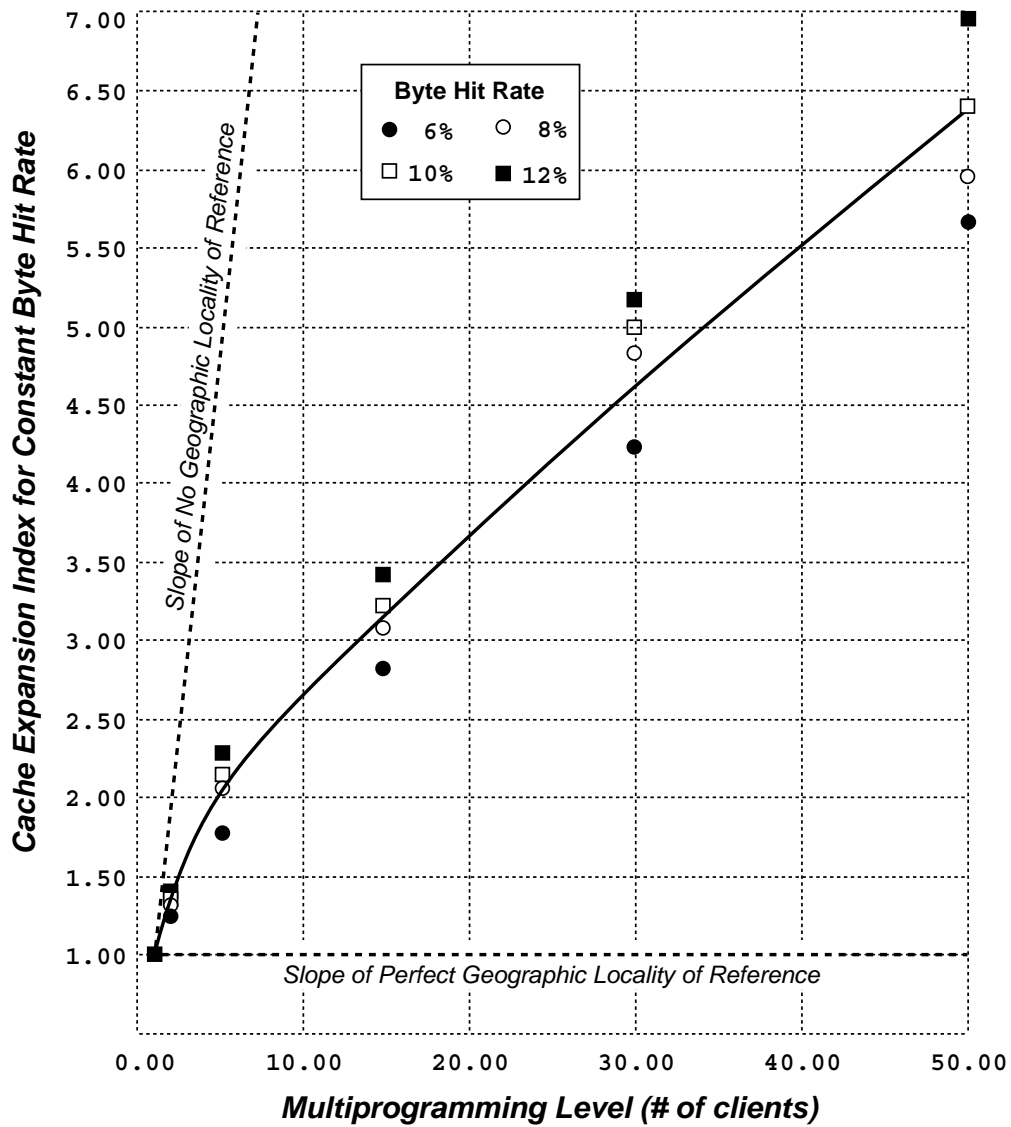
Sessions	4,700
Users	591
URLs Requested	575,775
Files Transferred	130,140
Unique Files Requested	46,830
Bytes Requested	2713 MB
Bytes Transferred	1849 MB
Unique Bytes Requested	1088 MB

Summary Statistics for Trace Data Used in This Study

Client-initiated Caching Effectiveness

Experiment Results

- ◇ Poor Byte Hit Rate < 40% with infinite cache.
- ◇ Sharing amongst multiple clients is limited too!



Cache Expansion Index for Remote documents

The Server's Perspective

Server Log Analysis

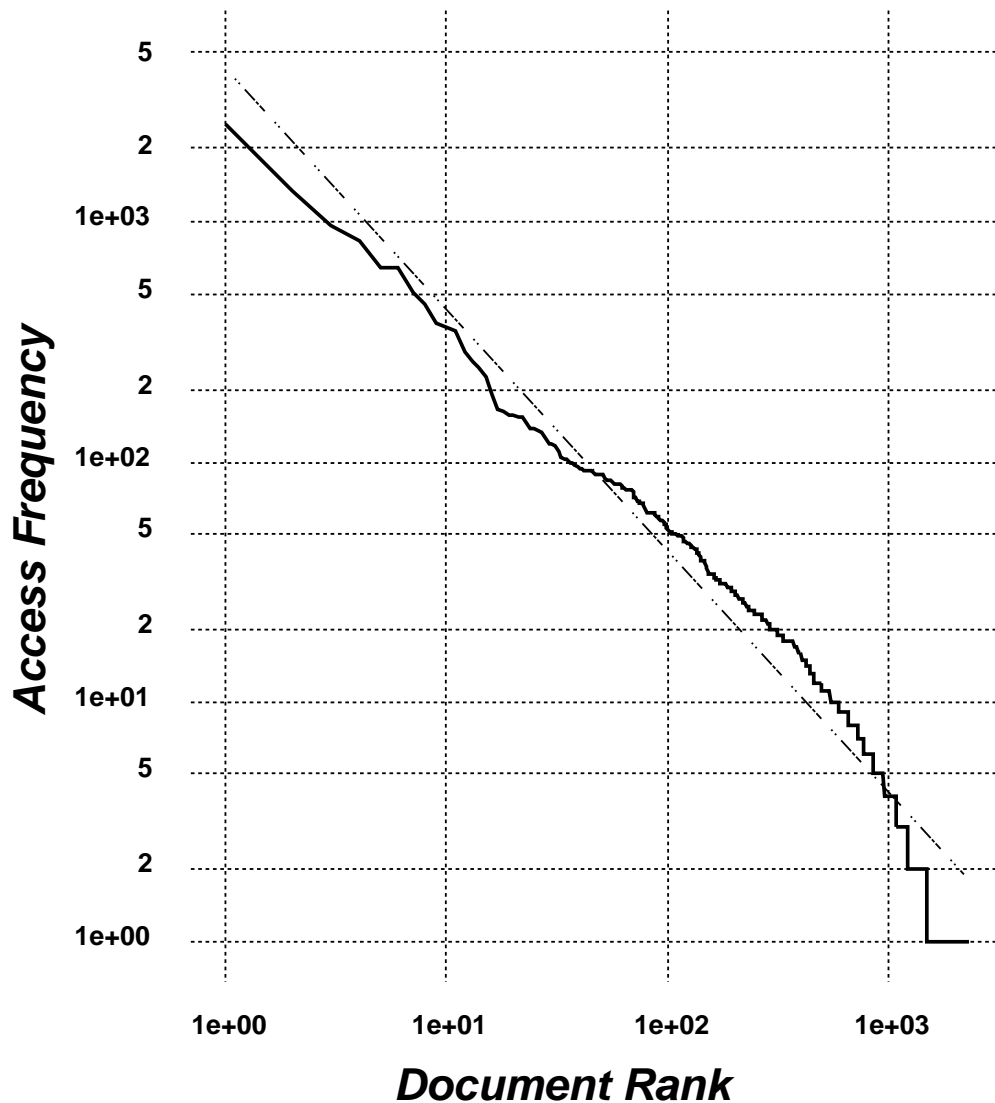
- ◇ We collected the logs of our departmental HTTP server and those of the Rolling Stones Multimedia server.
- ◇ We used the logs to analyze the popularity of various documents and to drive trace simulations of various server-initiated protocols.

	<code>cs-www.bu.edu</code>	<code>www.stones.edu</code>
Period	56 days	110 days
URL requests	172,635	4,068,432
Bytes transferred	1,447 MB	112,015 MB
Average daily transfer	26 MB	1,018 MB
Files on system	2,018	N/A
Files accessed (remotely)	974 (656)	N/A (1,151)
Size of (accessed) file system	50 MB (37 MB)	N/A (402 MB)
Unique clients (10+ requests)	8,123	60,461

Summary Statistics for Log Data Used in This Study

Popular documents are VERY popular!

Zipf's law governs the probability of document access



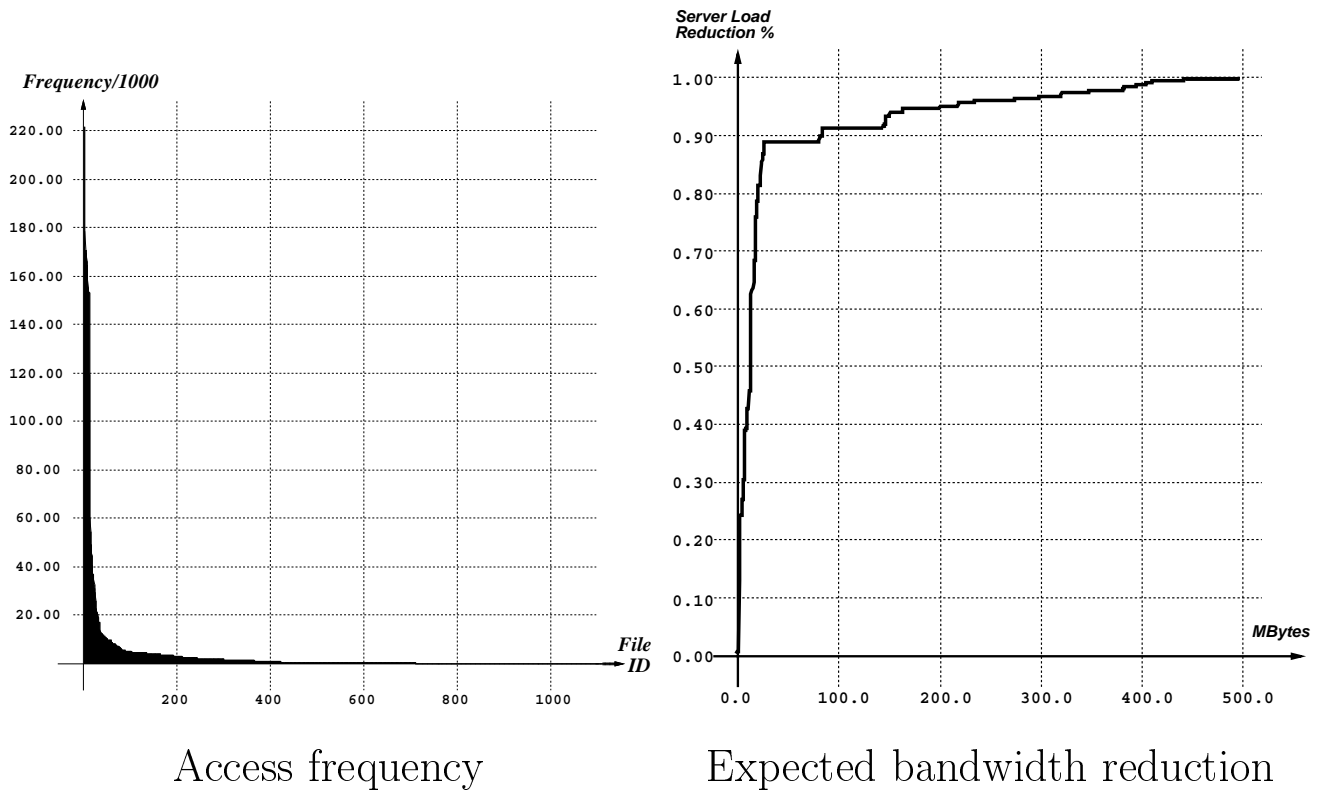
The popularity of a file is inversely proportional to the rank of that file

<http://cs-www.bu.edu>

The Server's Perspective

Log Analysis of <http://www.stones.com>

- ◇ Making 25MB ($\approx 3\%$) of data available to clients at a proxy one-hop closer to them would save more than 900MB/day of network bandwidth.

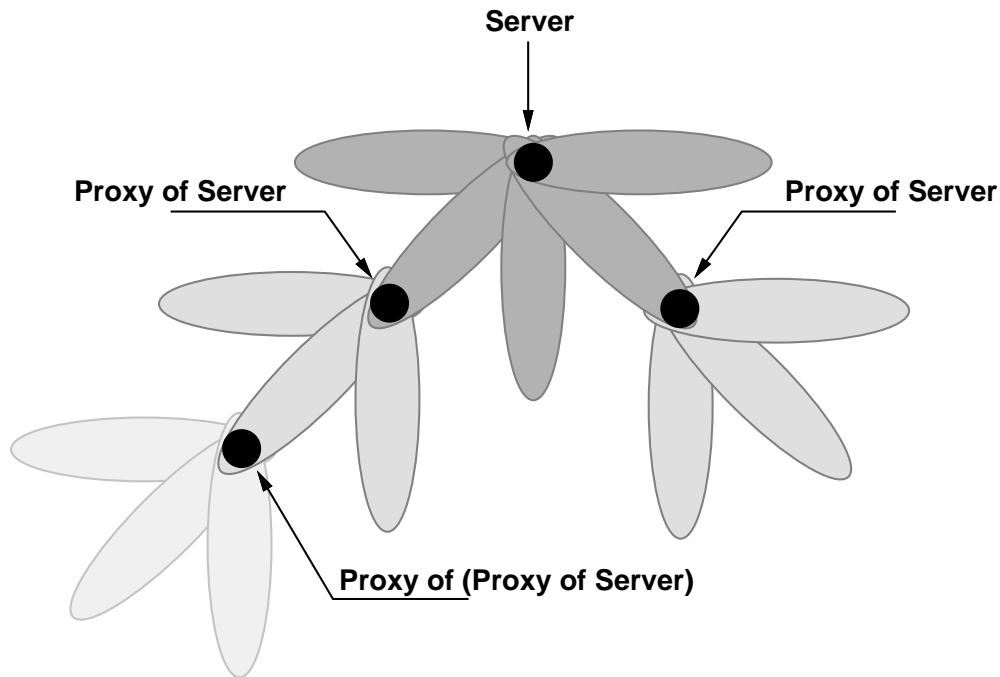


<http://www.stones.com>

Information Dissemination Protocol

Underlying Model

- ◇ A set of *service proxies* act as information “outlets” on the Internet.
- ◇ These *service proxies* offer space/bandwidth “for-rent” to other servers or proxies that constitute its *Cluster*.
- ◇ A server may belong to several clusters, thus allowing some of its files to be disseminated to multiple service proxies.
- ◇ Service proxies are themselves servers who may be members of other clusters.



Underlying Model for Information Dissemination

Information Dissemination Protocol

Questions to be answered

- ◇ Given the access pattern at a server, *which clusters should the server choose to join?*
- ◇ Given the access pattern at a server, *which files should the server disseminate? and where?*
- ◇ Given the popularity profile of all servers in a cluster, *how should the resources (space/bandwidth) at the service proxy be allocated?*

Assumptions

- ◇ The dissemination protocol should not require any “special” features/capabilities from other protocols.
- ◇ File popularity is a “universal” phenomenon (i.e. the probability of accessing a file is independent of who is accessing it). This is a *conservative* simplifying assumption.
- ◇ File popularity does not change drastically in a short period of time. This assumption has been verified.

Optimal Allocation of Storage at the Proxy

Notation

- ◇ $\mathcal{C} = \mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_n$ is the set of servers in a cluster. \mathcal{S}_0 is the service proxy of \mathcal{C} .
- ◇ R_i is the total number of bytes per unit time serviced by server \mathcal{S}_i to clients outside \mathcal{C} .
- ◇ $H_i(b)$ is the probability that a request to \mathcal{S}_i will be to the most popular b bytes disseminated to \mathcal{S}_0 .
- ◇ B_i is the number of bytes that \mathcal{S}_0 duplicates from \mathcal{S}_i . $B_0 = B_1 + B_2 + \dots + B_n$ is the total storage space available at \mathcal{S}_0 .

Goal

- ◇ Choose B_i to maximize the percentage of traffic serviced at \mathcal{S}_0 .

$$\alpha_{\mathcal{C}} = \frac{\sum_{i=1}^n R_i \times H_i(B_i)}{\sum_{i=1}^n R_i}$$

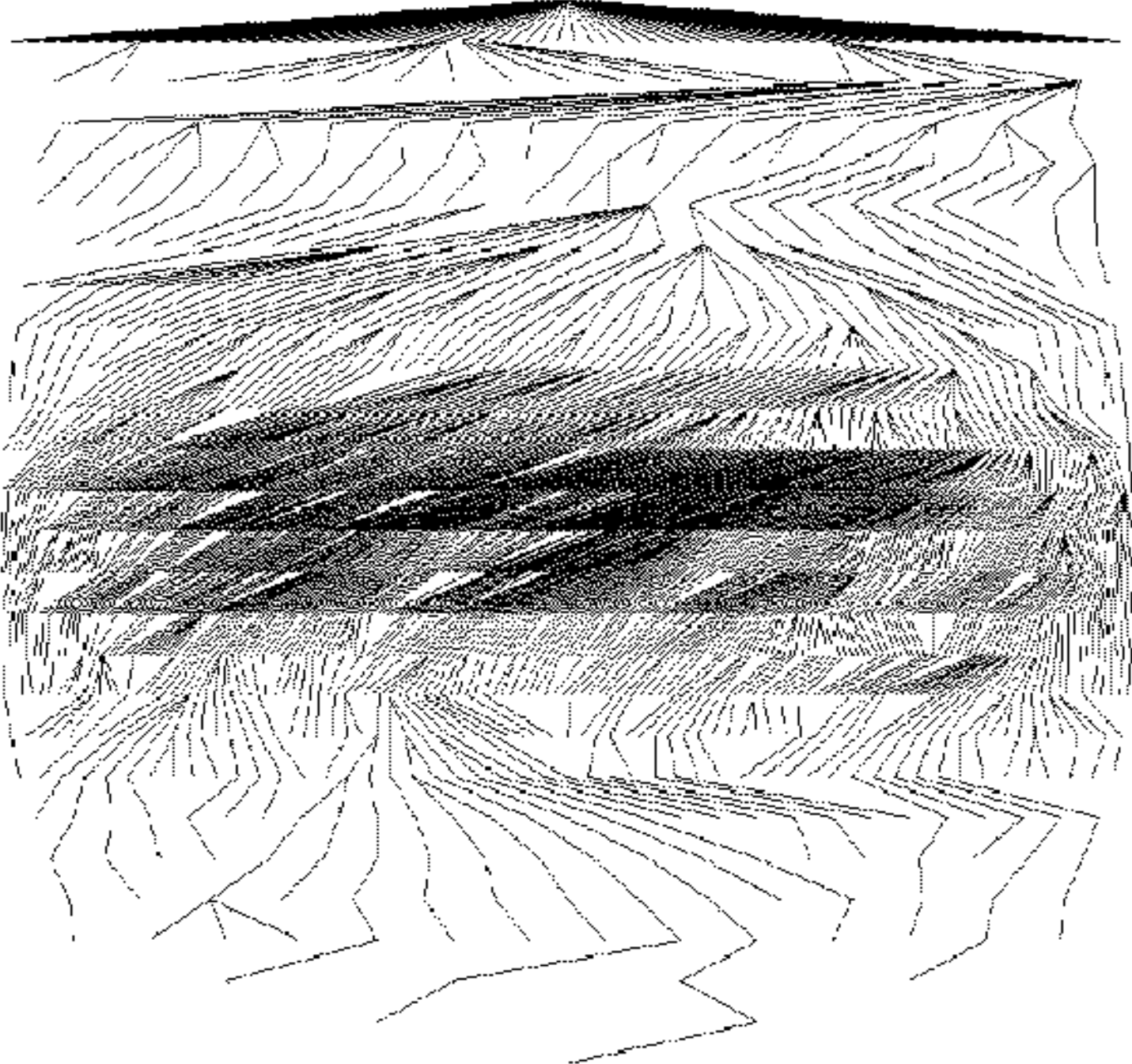
Which Proxies Should be Contracted?

Characterizing the Client Tree and Choosing Proxies

- ◇ Using the **record route** option of TCP/IP, it is possible to build a complete tree originating at the server with clients at the leaves. For **http://cs-www.bu.edu**, this tree consisted of 18,000 nodes.
- ◇ The most popular files are disseminated down the tree and stored at proxies *closer* to the clients.
- ◇ The location of such proxies depends on the demand from the various parts of the tree.
- ◇ Analysis of **http://cs-www.bu.edu** logs for a consecutive 26-week period suggests that the shape of the tree (especially internal nodes) and the distribution of load is quite static over time.

How Does the Internet Look to a Server?

Server

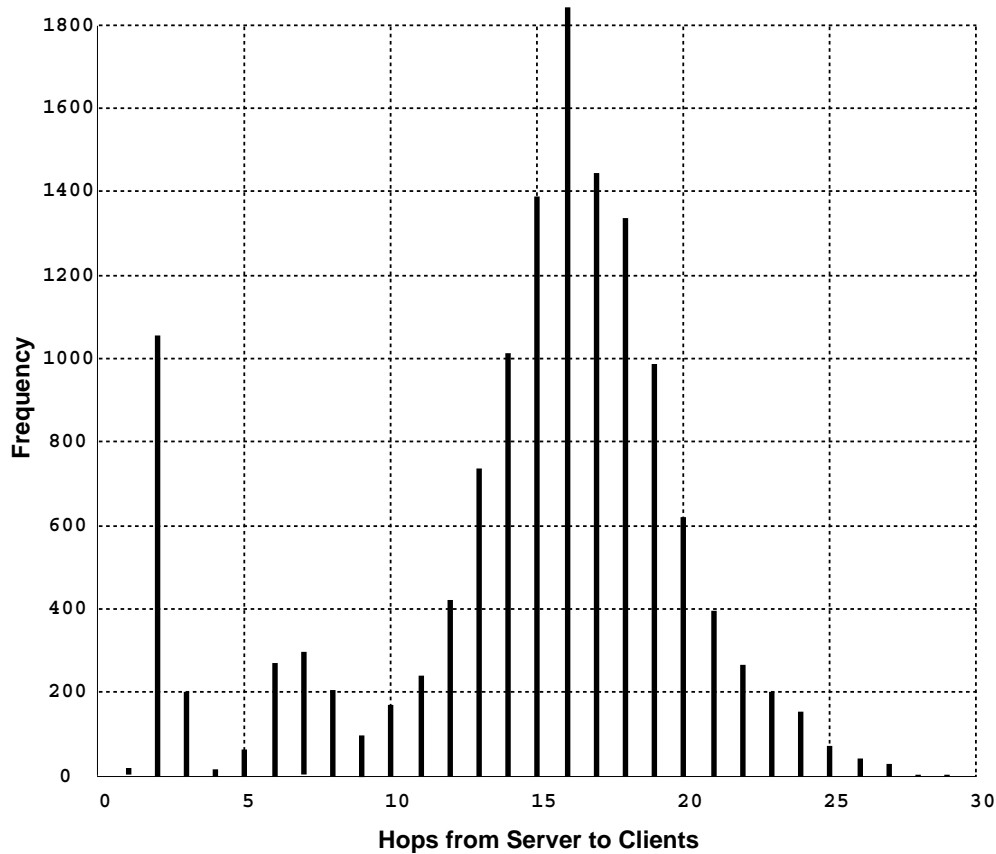


Clients

How Much Bandwidth is Saved?

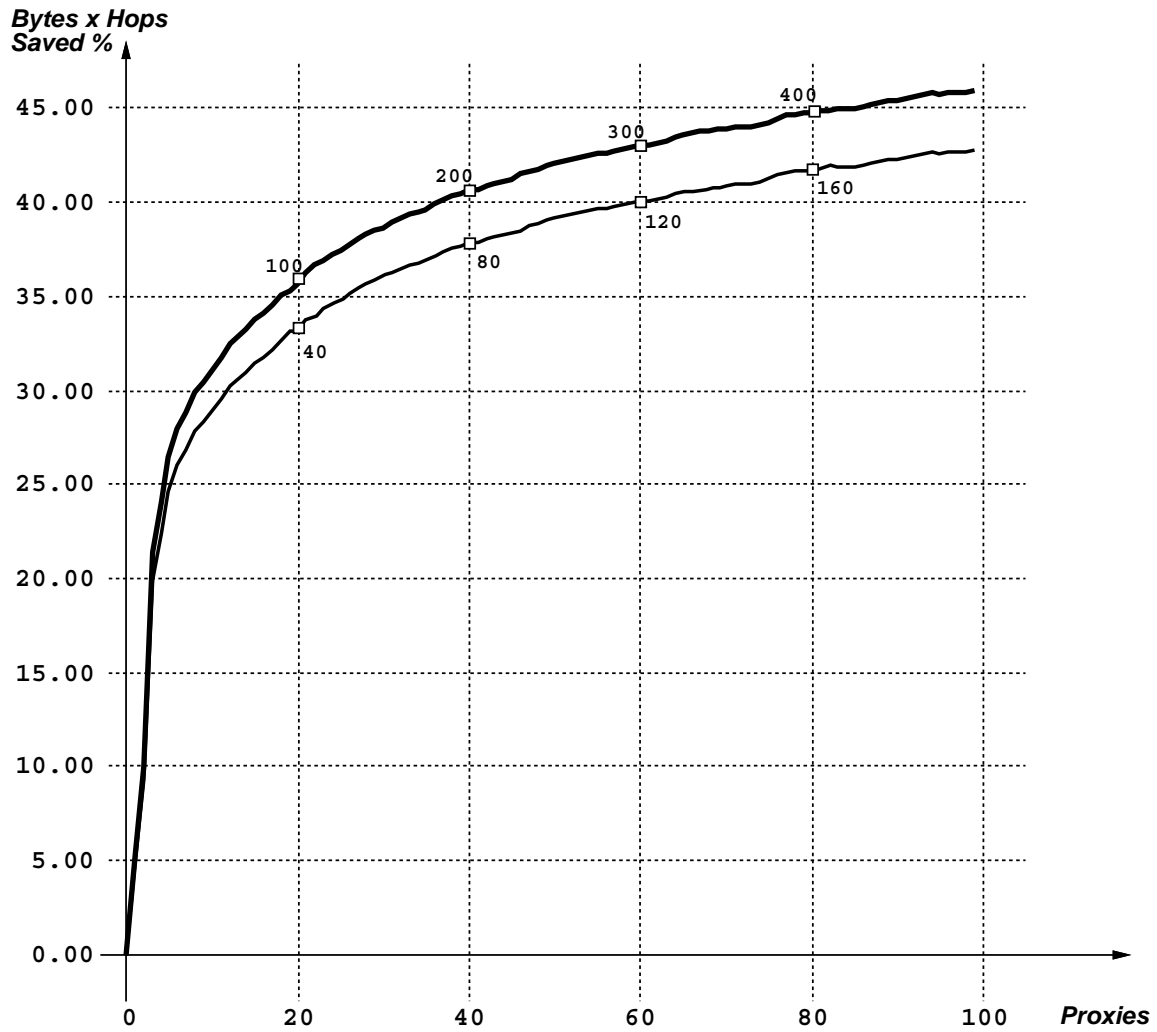
How far could we “push” information towards clients?

- ◇ At least 8-9 hops!
- ◇ Replicating the most popular 25 MB from `http://www.stones.com` on *few* proxies yields a whopping saving of > 8 GBHops of network bandwidth per day.



How far away are clients?

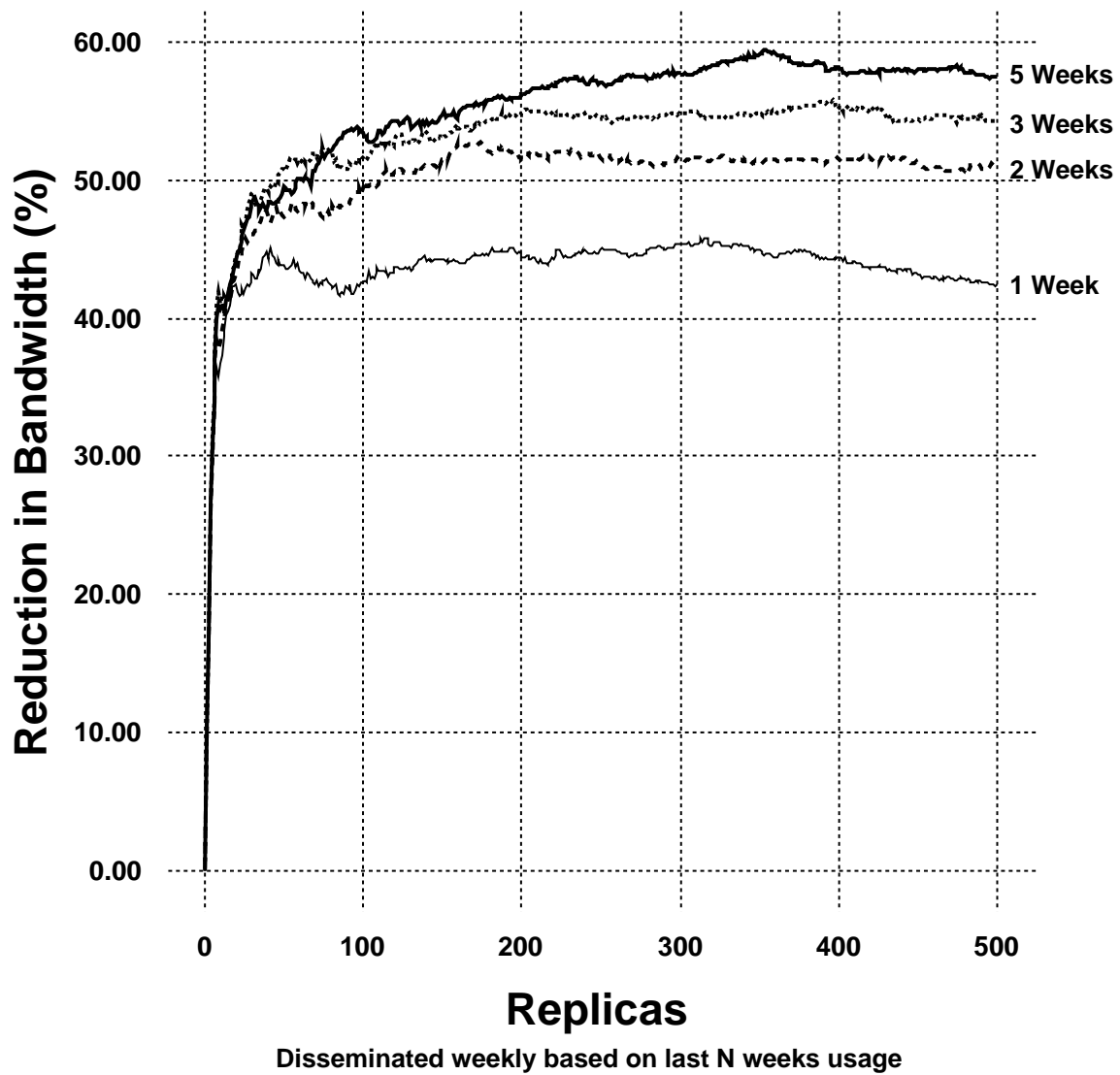
How Much Bandwidth is Saved?



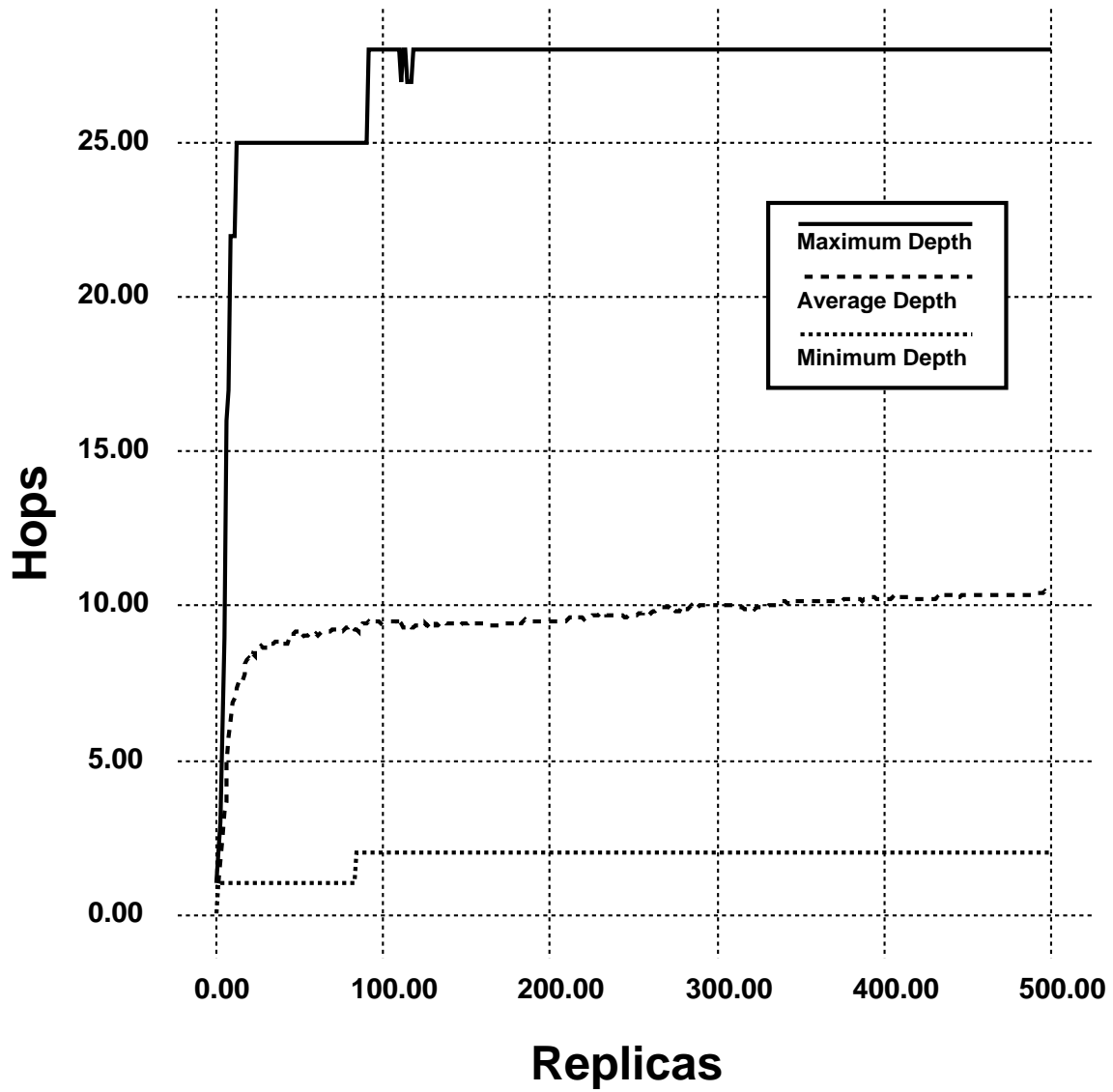
Expected reduction in bandwidth as a result of dissemination

How Much History Should be Used to Disseminate?

- ◇ Short histories are better for shallow dissemination.
- ◇ Long histories are better for deep dissemination.

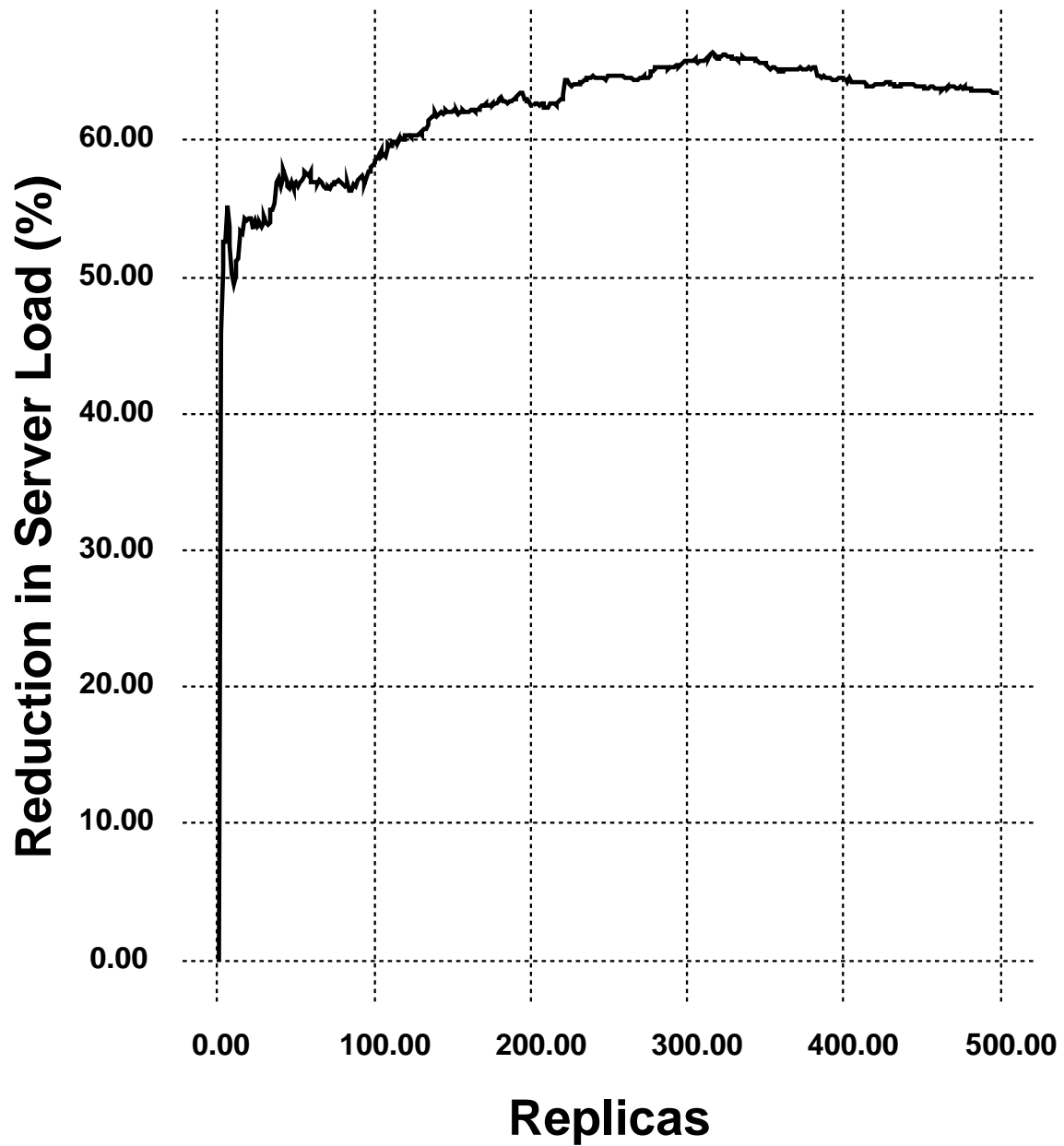


Dissemination Depth



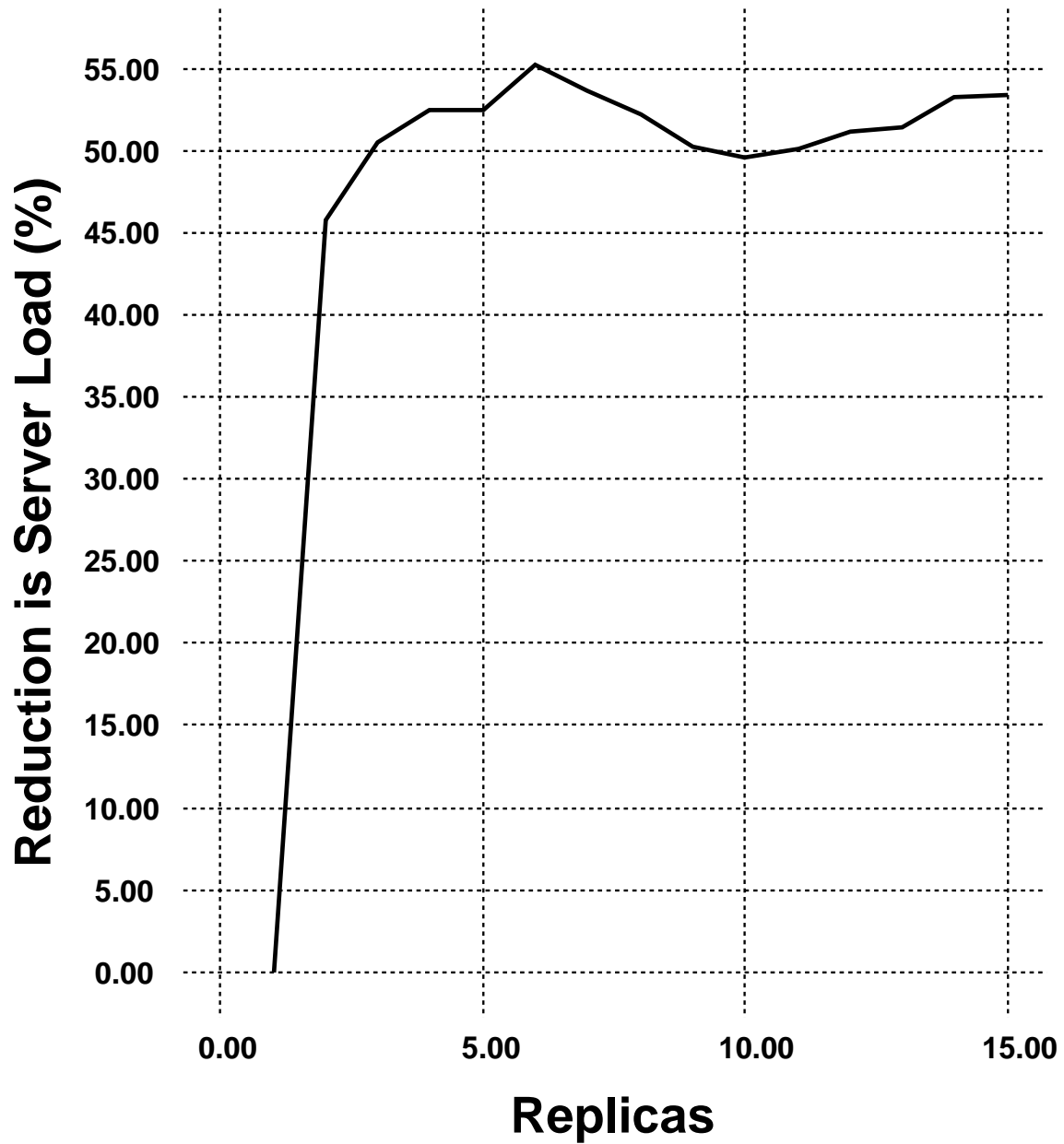
How far down the tree does dissemination go?

How Much Reduction in Server Load?



Expected reduction in server load as a result of dissemination

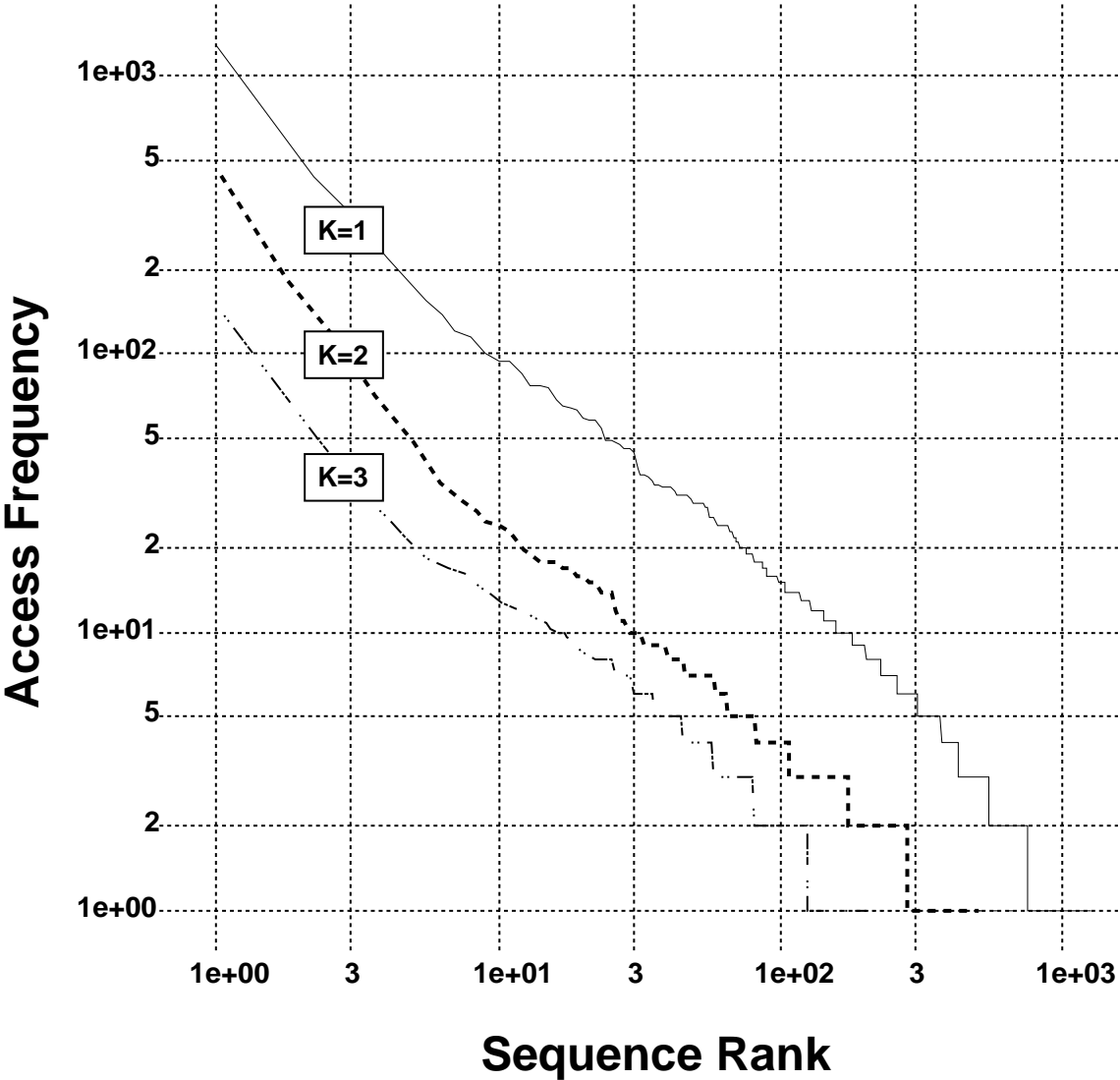
How Much Reduction in Server Load? (closeup)



Expected reduction in server load as a result of dissemination

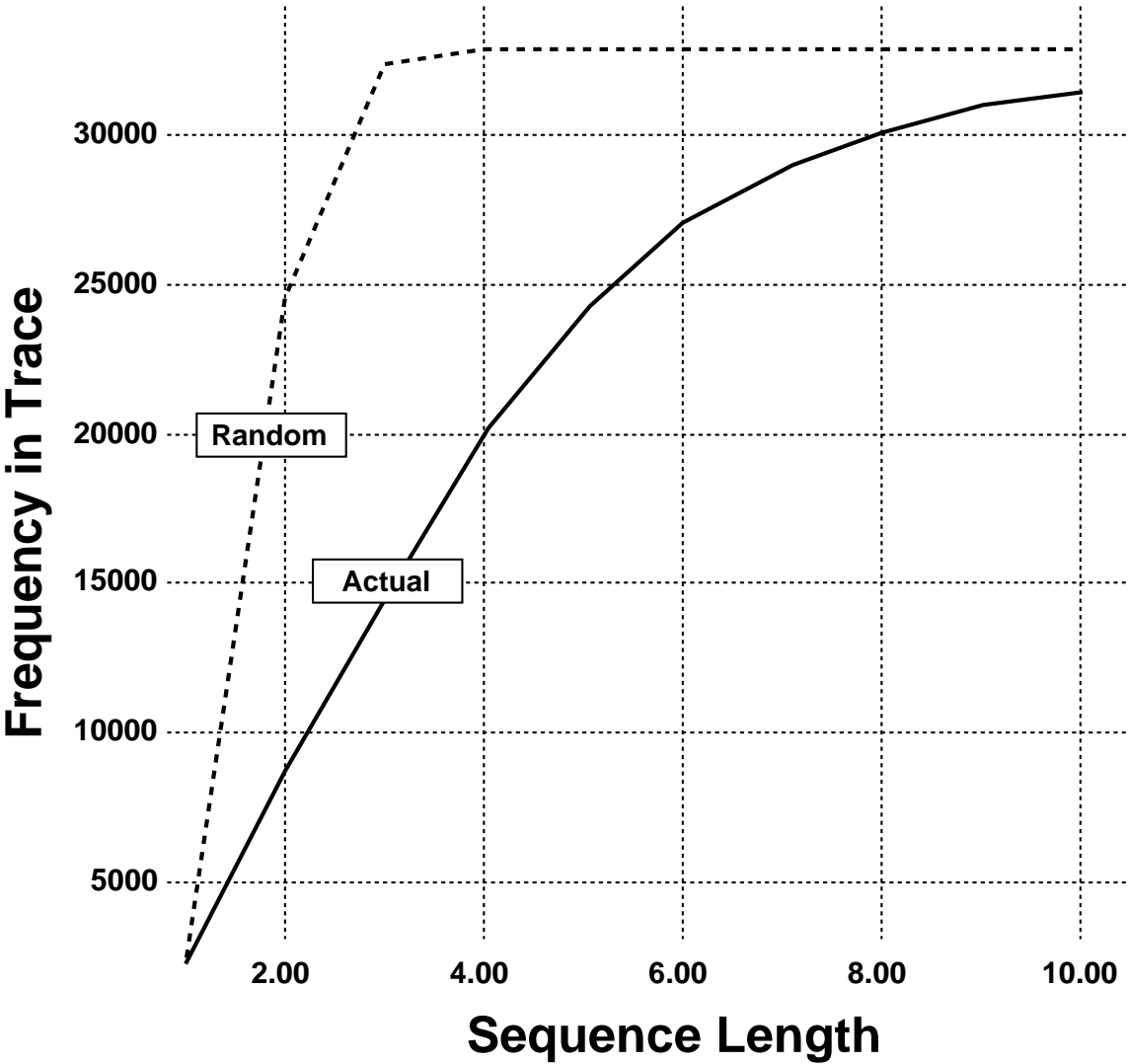
Spatial Locality of reference

◇ Zipf's law governs the probability of sequences of document access! The popularity of a sequence of file requests is inversely proportional to the rank of that sequence.



Spatial Locality of reference

There are far less sequences in a server trace than randomly possible!

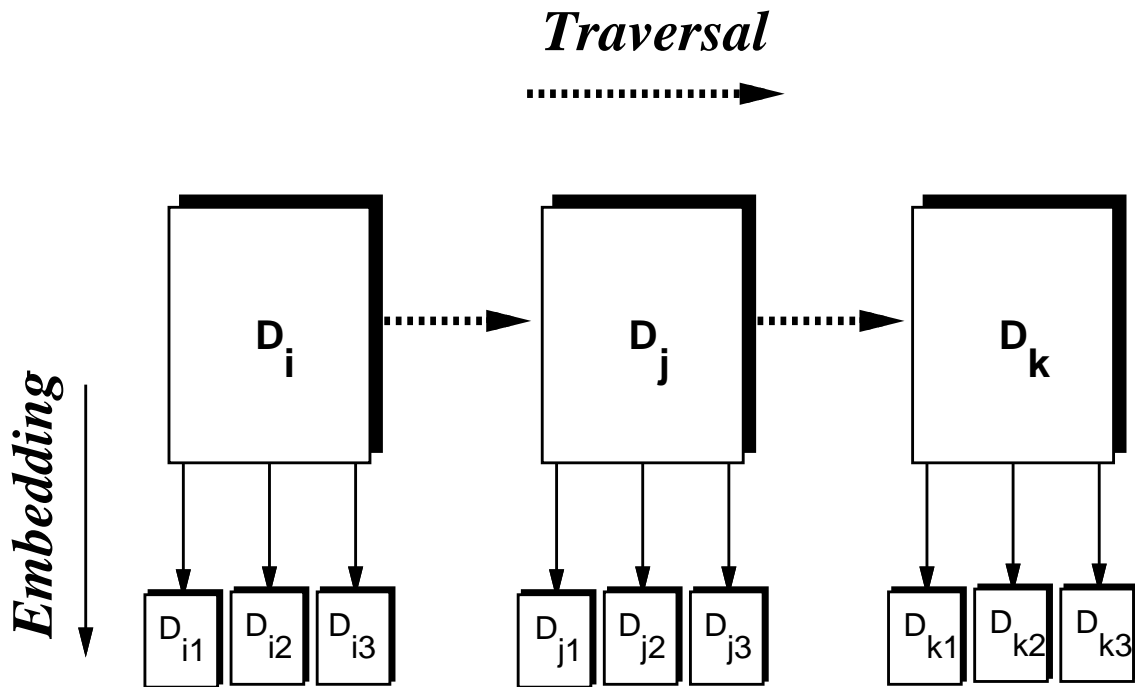


Could one use the knowledge of “popular” sequences for prefetching?

The Notion of Speculative Service

Could client requests be predicted by the server?

- ◇ In many cases, the answer is *yes*.
- ◇ Servers could “speculatively” service documents *before* they are requested (a.k.a. server-initiated prefetching).



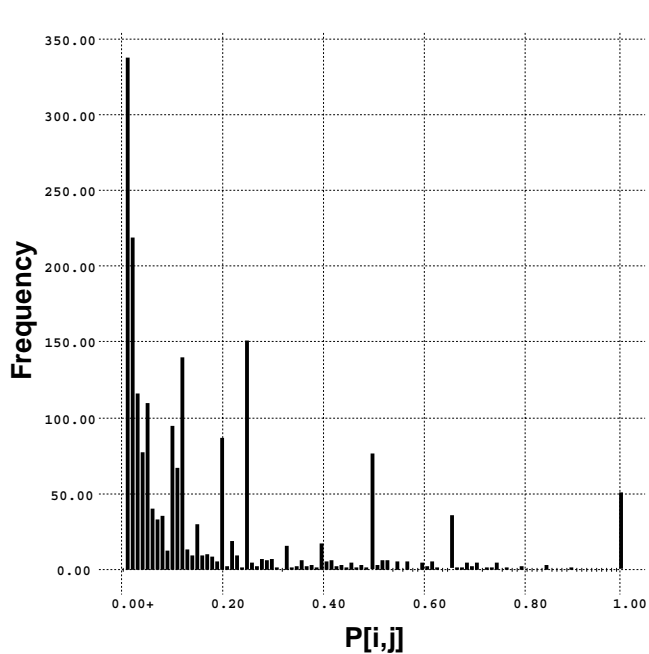
Document Access Interdependency Matrix

Notation

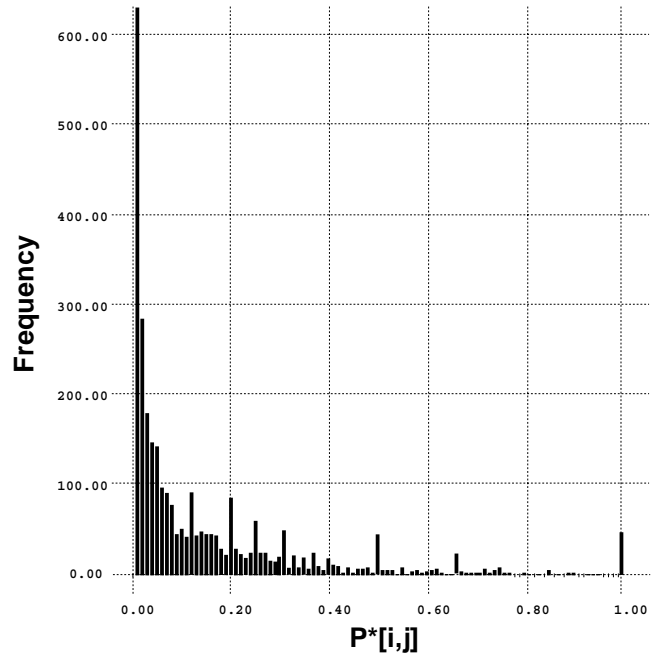
- ◇ Let $p[i, j]$ denote the conditional probability that document \mathcal{D}_j will be requested, within T_w units of time after the request for \mathcal{D}_i .
- ◇ Let P denote the square matrix $p[i, j]$, for all possible documents $0 \leq i, j \leq N$. Let P^* denote the transitive closure of P .
- ◇ Thus, $p^*[i, j]$ is the probability that there will be a sequence of requests (inter-request time $\leq T_w$) starting with \mathcal{D}_i and ending with \mathcal{D}_j .

Server log analysis

- ◇ Using server logs, the P and P^* matrices could be easily constructed.



Histogram for ranges of $p[i, j]$



Histogram for ranges of $p^*[i, j]$

Speculative Service Experiments

Simulation Model

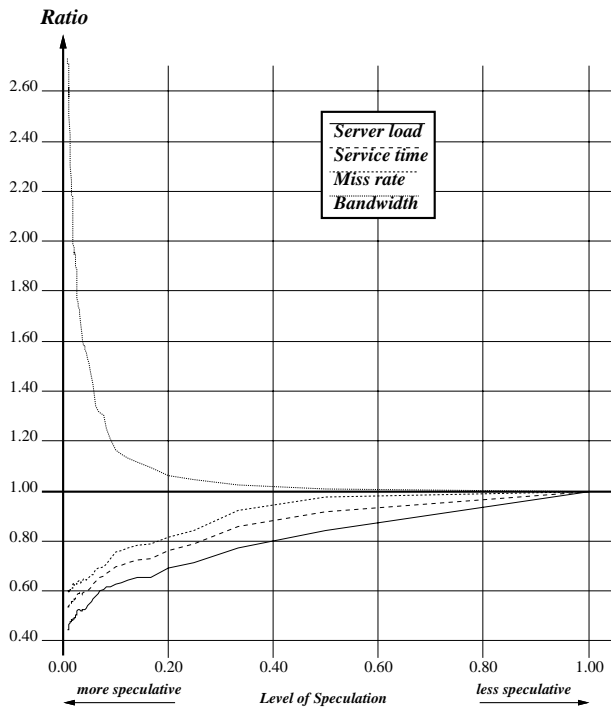
- ◇ Successive requests separated by less than **StrideTimeout** units of time belong to the same “stride”.
- ◇ Clients maintain a session cache. The session cache is purged if the time between successive requests exceeds **SessionTimeout**.
- ◇ Four performance metrics are used: **Bandwidth ratio**, **Server Load ratio**, **Service Time ratio**, and **Miss rate ratio**.

Parameter	Meaning	Base Value
CommCost	Cost of communicating 1 Byte	1 unit
ServCost	Setup cost for a service request	10,000 unit
StrideTimeout	Value of time window T_w	5.0 secs
SessionTimeout	Cache invalidation timeout	∞ secs
MaxSize	Maximum size to prefetch	∞ (no limit)
Policy	Speculative service algorithm	$p^*[i, j] \geq T_p$
HistoryLength	Length of the logs used for P	60 days
UpdateCycle	Frequency of recomputing P	1 day

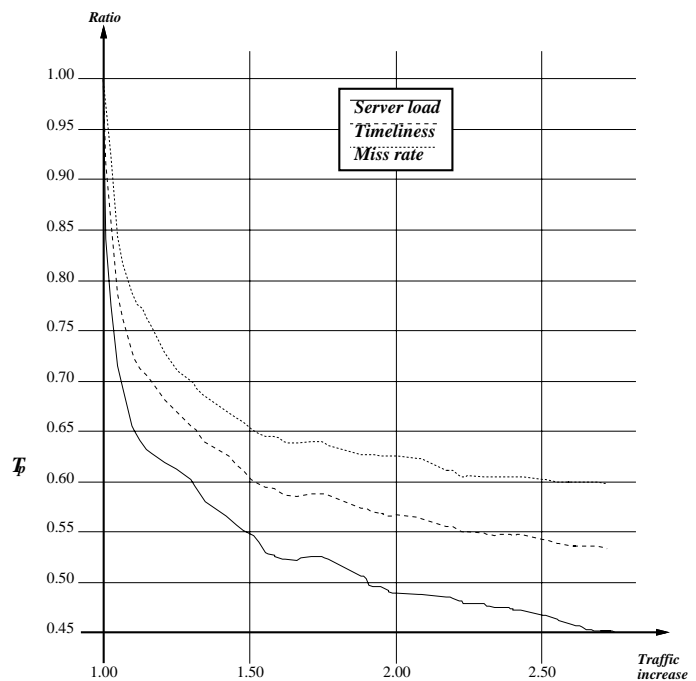
Speculative Service Experiments

Baseline Results

- ◇ Significant improvement in performance (above what is achievable by client caching) could be achieved for a miniscule increase in traffic.
- ◇ 5% extra bandwidth results in a whopping 30% reduction in server load, a 23% reduction in service time, and a 18% reduction in client miss-rate.
- ◇ Beyond some point, speculation does not seem to pay off.



Metrics vs Speculation Level

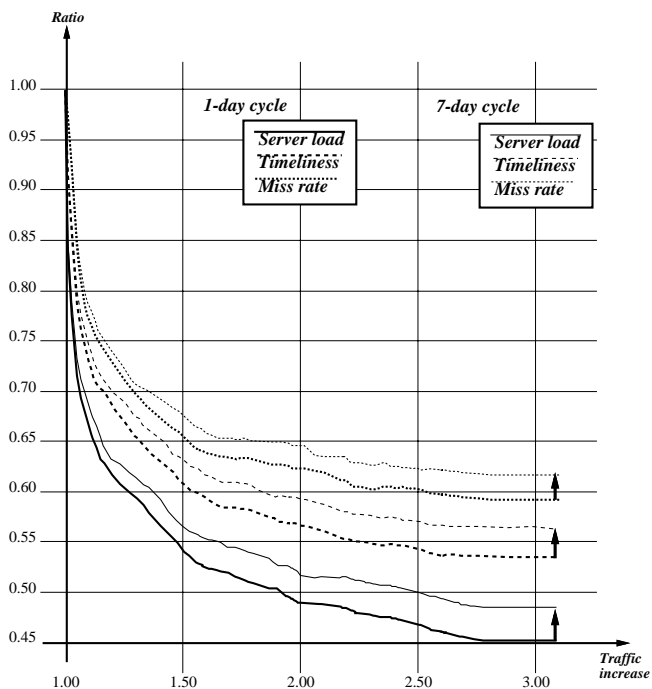


Metrics vs Bandwidth

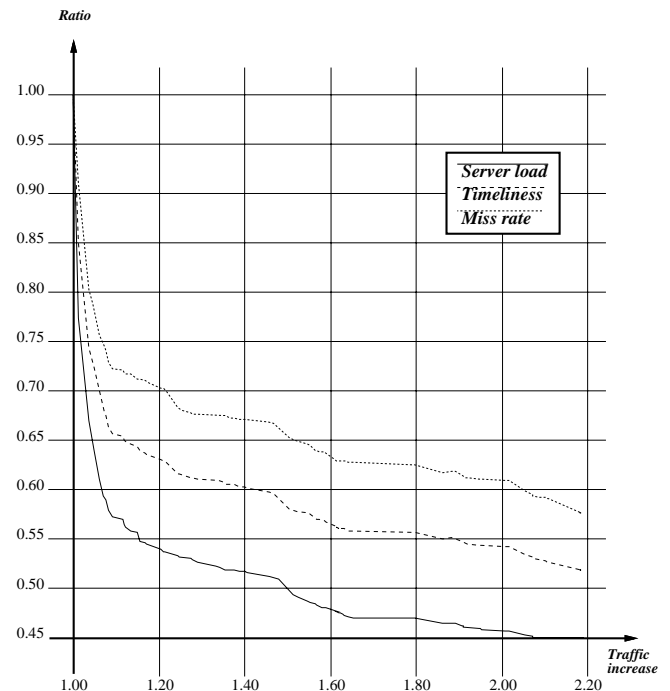
Speculative Service Experiments

Stability of the P and P^* Relations

- ◇ We varied the `UpdateCycle` from 1 to 7 days, while keeping the `HistoryLength` at 60 days. This change resulted in a 3% degradation in all measured metrics, suggesting that P and P^* do change (albeit very slowly) with time.
- ◇ Also, we varied the `HistoryLength` from 60 to 30 days, while keeping the `UpdateCycle` at 1 day. This change resulted in a 5% improvement in all measured metrics, suggesting that an aging mechanism must be used to phase-out dependencies exhibited in on older server traces, in favor of dependencies exhibited in more recent ones.



Effect of slower `UpdateCycle`

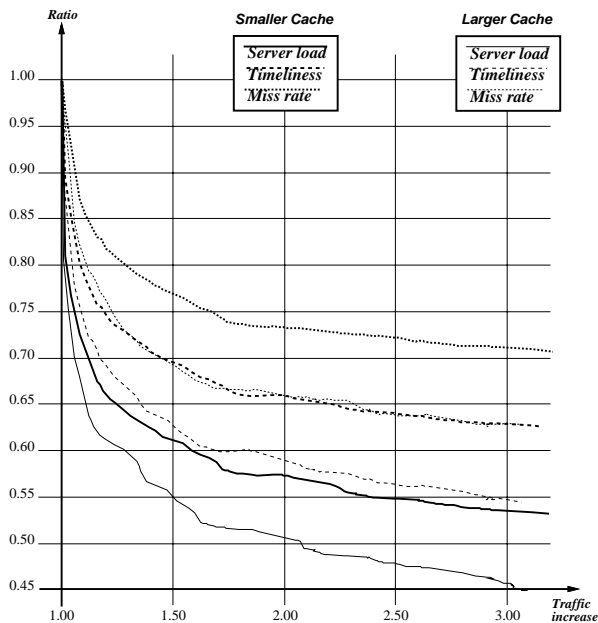


Effect of shorter `HistoryLength`

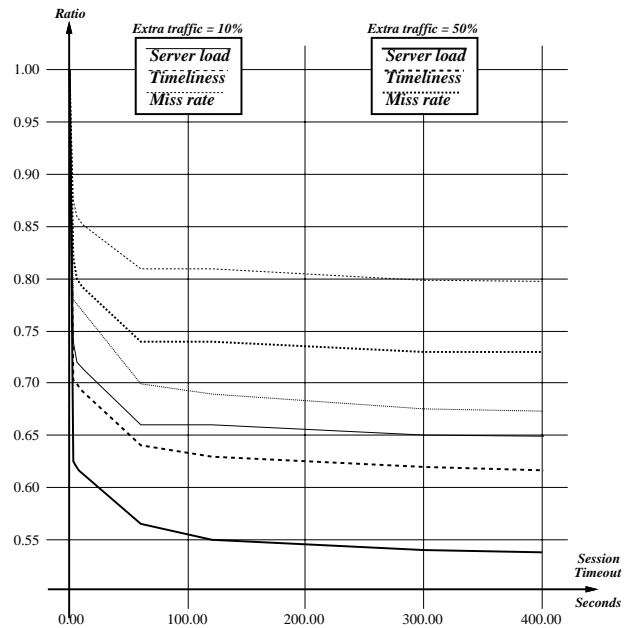
Speculative Service Experiments

Effect of Client Caching

- ◇ We compared simulations with `SessionTimeout` equal to 3,600 seconds (large cache) and to 120 seconds (small cache).
- ◇ The presence of client caching (even if modest) is likely to further improve the performance of speculative service.
- ◇ In order to reap all the benefit from speculative service, client must cache “prefetched” documents long enough.



Effect of client caching

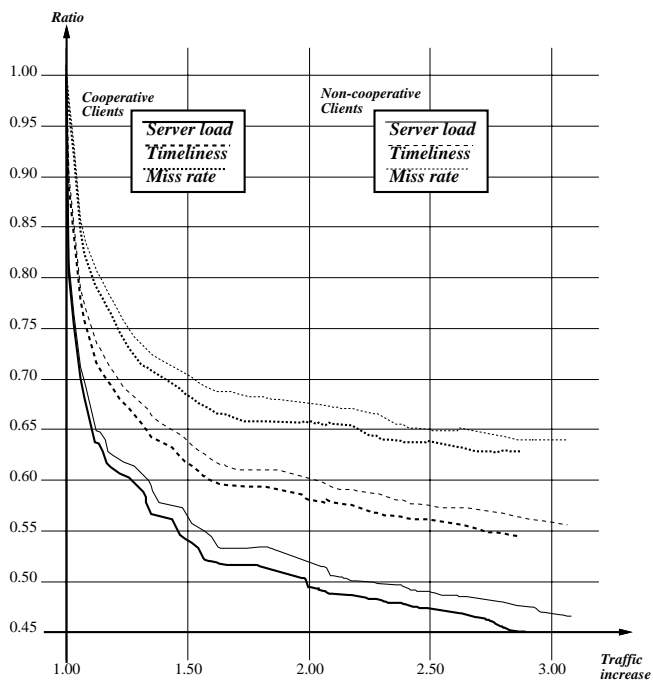


Efficiency of client caching

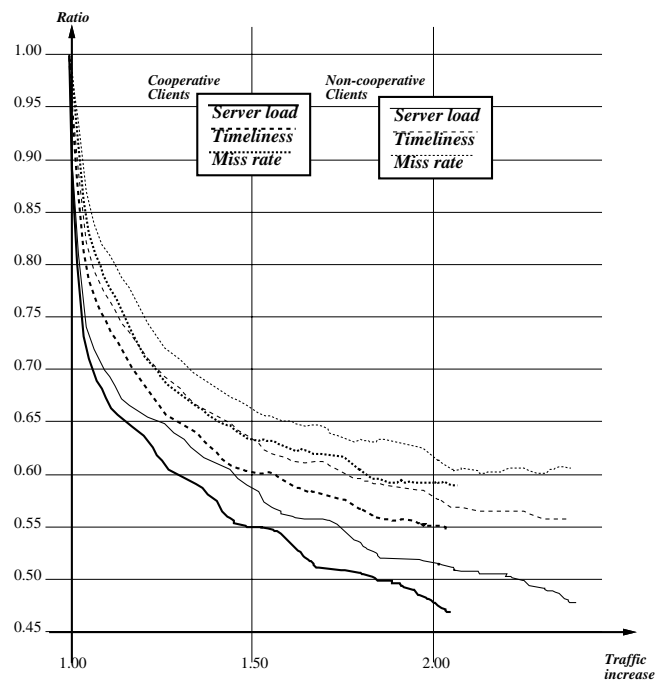
Speculative Service Experiments

Cooperative Clients

- ◇ Performance could be further improved if documents “already cached at the client” are not speculatively serviced!
- ◇ Our simulations showed that speculative service with cooperative clients results in better bandwidth utilization, especially when the client performs “some” caching.



Cooperative client (small cache)

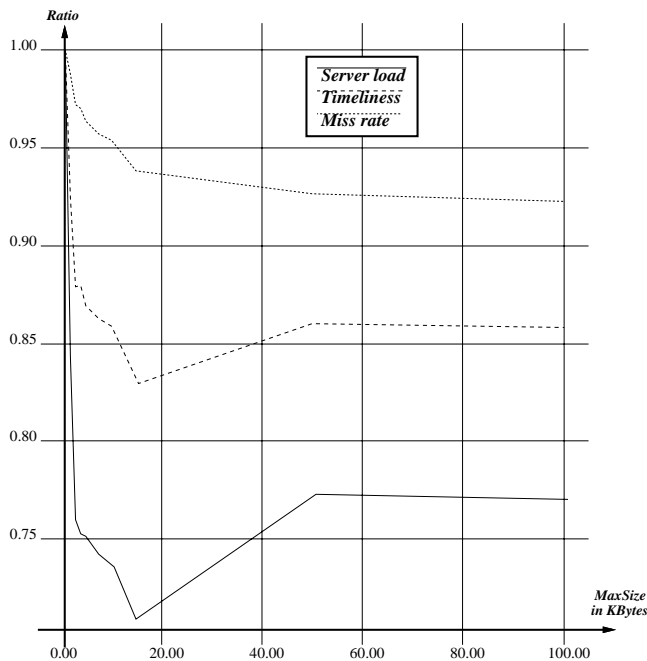


Cooperative client (infinite cache)

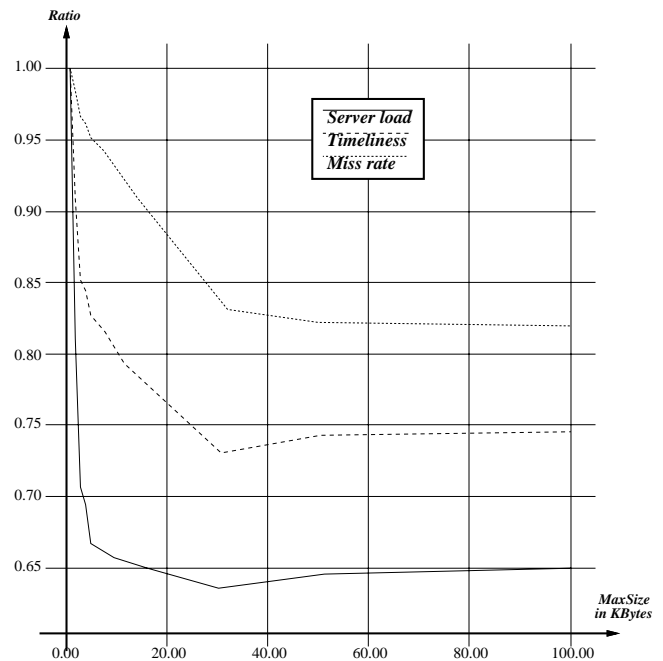
Speculative Service Experiments

Effect of Document Size

- ◇ The benefits of speculation are most pronounced when documents serviced speculatively are small. We studied this by varying the `MaxSize` parameter.



Low level of speculation

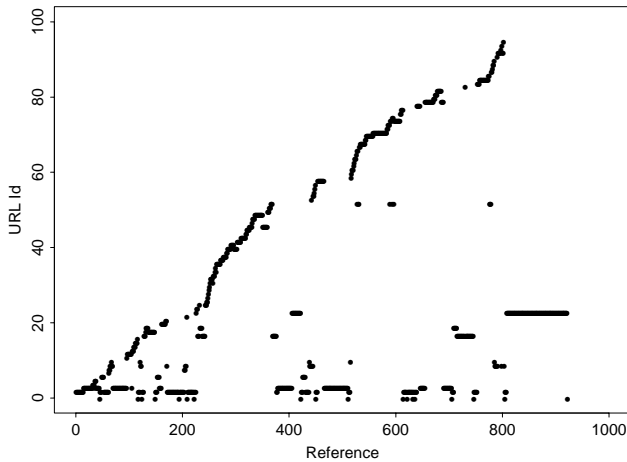


High level of speculation

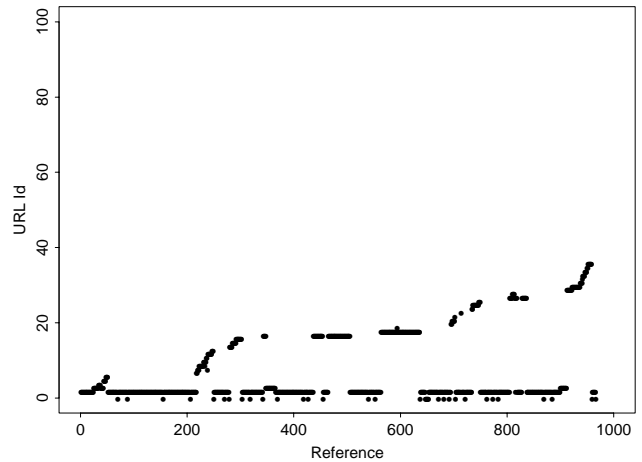
Client Prefetching

Could the next request be predicted by the client?

- ◇ Using client traces, it is possible for the client software to perform “prefetching” [Bestavros and Cunha: 1995].
- ◇ Performance of client-initiated prefetching is highly dependent on user access patterns.



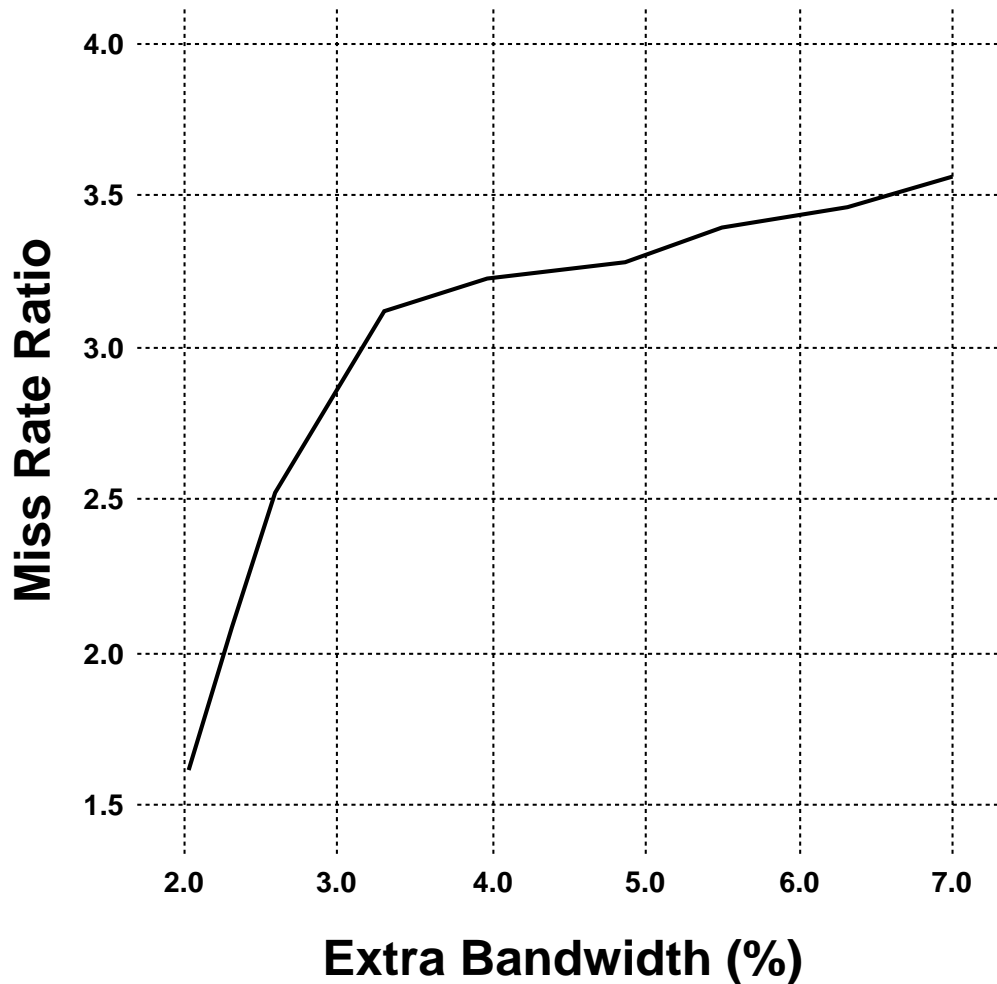
A “Mostly Surfing” User



A “Mostly Working” User

Performance of Client Prefetching

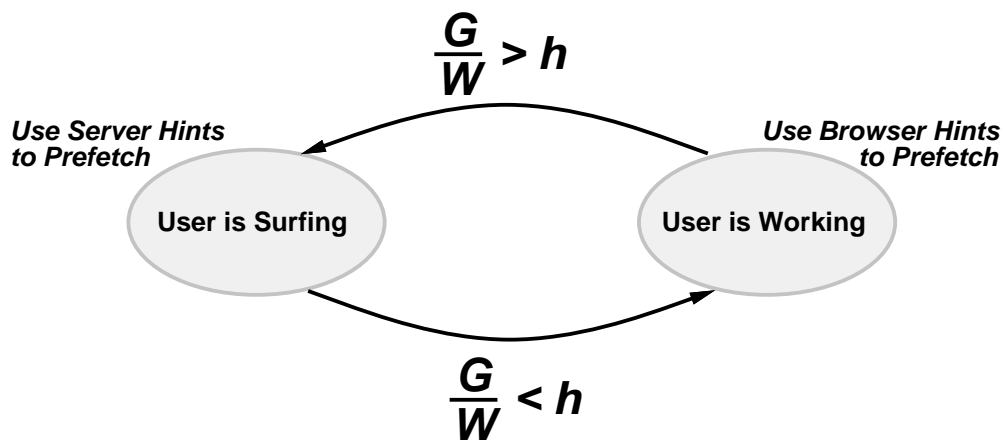
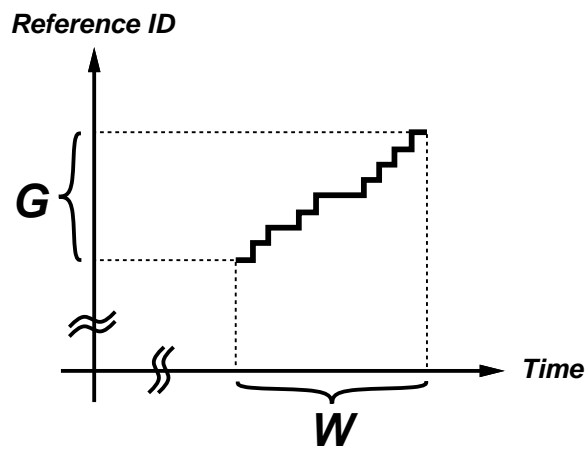
- Trace simulations indicate that client-initiated prefetching is *very* effective for “frequently traversed documents” but ineffective for “newly/rarely traversed documents”.



Performance improvement for a “creature of habit” 😊

Flip-Flop Between Server and Browser Hints

- By monitoring the “surfing index”, we favor either the server hints or the browser hints for prefetching purposes.



Implementation of Prefetching Agents

Server-assisted Prefetching

- We have modified the NCSA HTTP server to allow it to build first and second Markov models for all documents on a site.
- Everytime a client requests a page, the server automatically appends **PREFETCH** hints to the end of the served document:

```
<PREFETCH AGENT="agent" HREF="url" RANK="rank%" PROB="prob">
```
- Also, we have modified the NCSA Mosaic browser to allow it to interpret the server hints and perform prefetching accordingly.

Client-initiated Prefetching

- We have modified the NCSA Mosaic browser to allow it to build first and second Markov models for all documents accessed by a user.
- Everytime a client requests a document, the browser (may) automatically prefetch a number of other documents and keep them in the disk cache.

Integrating Server and Client agents

- The above two implementations need to be integrated. We are working on that!

Web Traffic Characteristics

Motive

- Traffic attributed to the use of the Web makes up an increasing percentage of internet traffic.
- Understanding the nature of web traffic is important for scaling and capacity planning purposes.

Conclusions

- Traffic attributed to the use of the Web exhibits self-similar properties.
- Web traffic self-similarity is due to the heavy-tailed nature of transmission time distribution, which is related to Web file size distributions.

Related Work

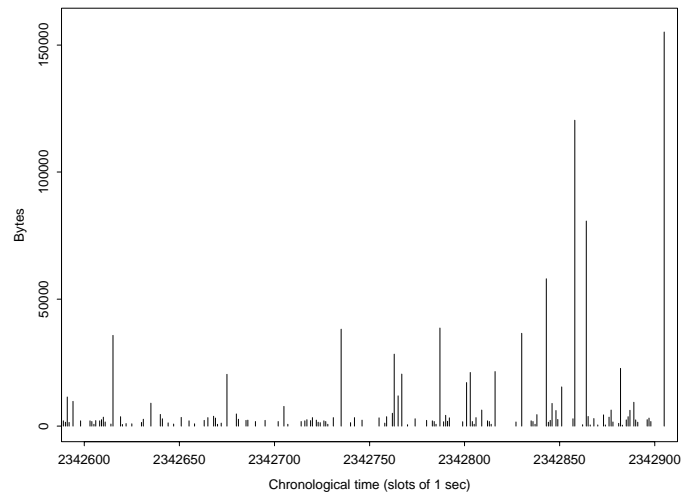
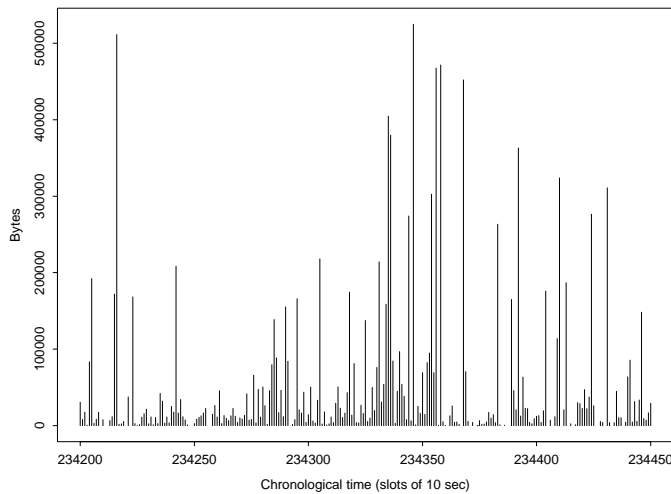
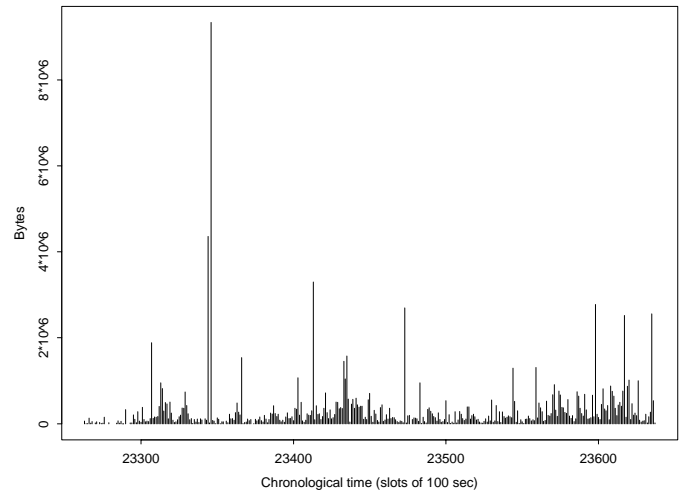
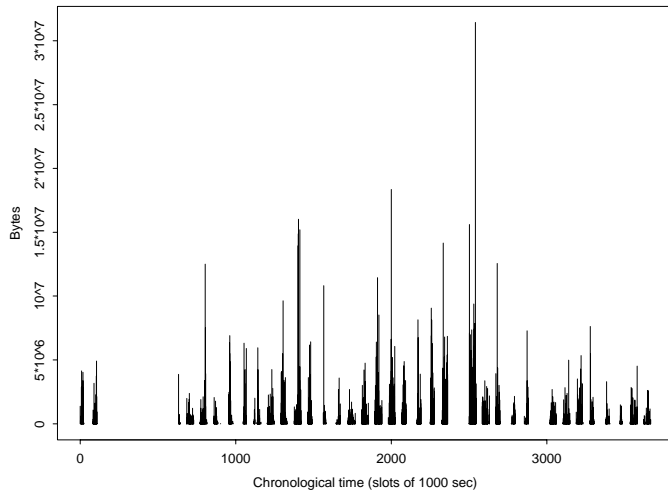
- Leland, Taqqu, Willinger, and Wilson's paper in SIGCOMM'93 established the self-similarity of LAN traffic and Willinger, Taqqu, Sherman, and Wilson's paper in SIGCOMM'95 explained self-similarity for LAN using ON/OFF model.

Web Traffic Characteristics: Data Set used in this study

Sessions	4,700
Users	591
URLs Requested	575,775
Files Transferred	130,140
Unique Files Requested	46,830
Bytes Requested	2713 MB
Bytes Transferred	1849 MB
Unique Bytes Requested	1088 MB

Summary Statistics for Trace Data Used in This Study

Web Traffic Characteristics: Burstiness at varying scales



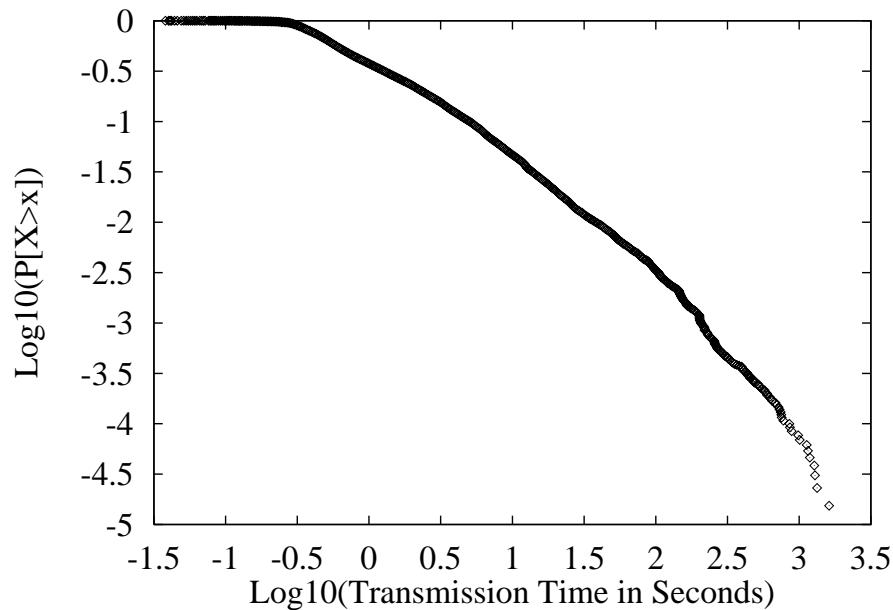
Traffic Bursts over Four Orders of Magnitude

Upper Left: 1000, Upper Right: 100, Lower Left: 10, and Lower Right: 1 Second Aggegrations.

Web Traffic Characteristics: Transmission Time Distribution

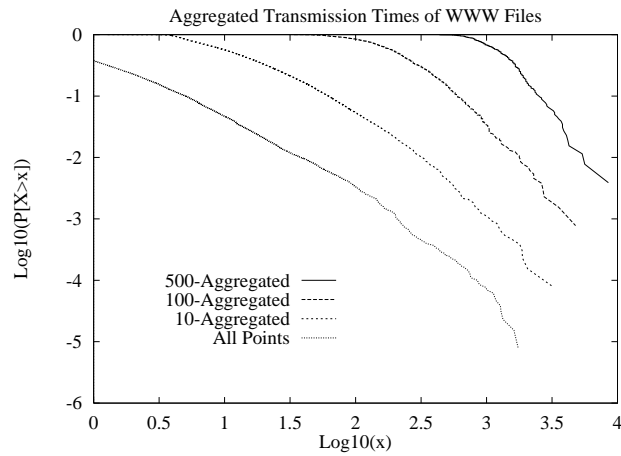
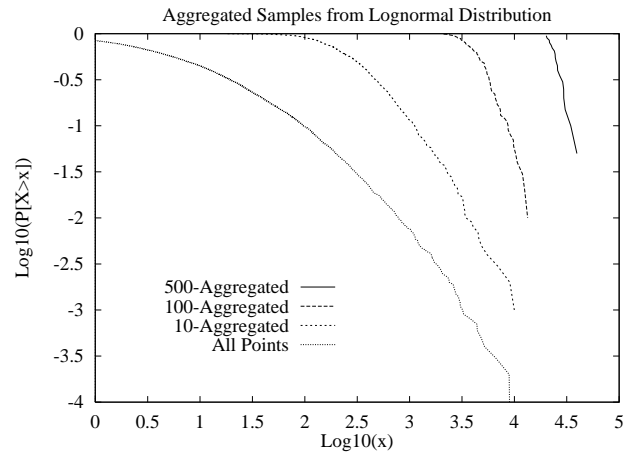
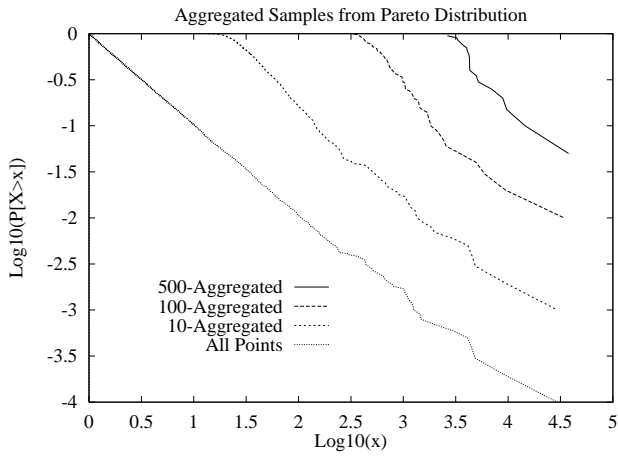
- Web traffic self similarity may be the result of superimposing many ON/OFF processes, where ON and/or OFF times are heavy-tailed (Pareto).

$$p(x) = \alpha k^\alpha x^{-\alpha-1}$$



- The fit to a straight line of the Log-Log Complementary Distribution (LLCD) suggests that transmission times are heavy-tailed (*i.e.* Pareto with parameter α).

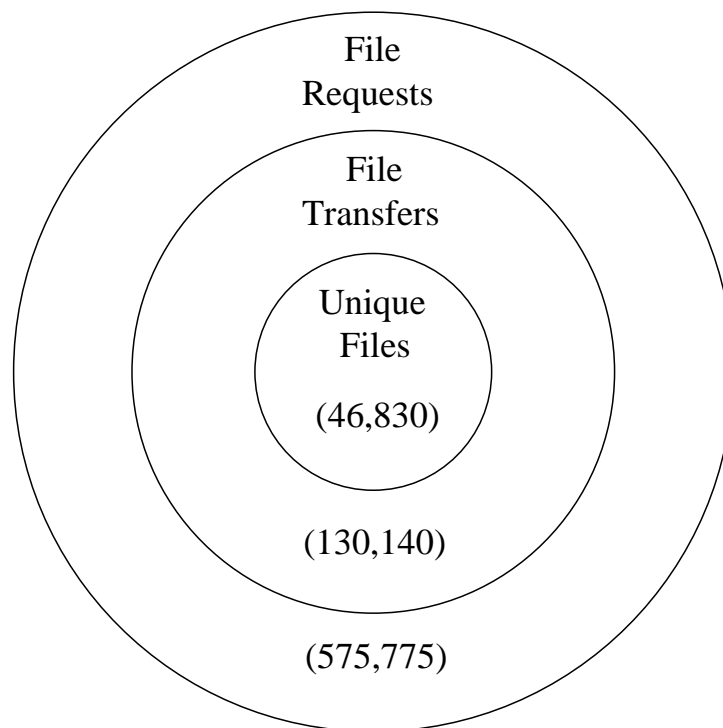
Heavy-tailed Transmission Times: The CLT Test



Comparison of CLT Test for Pareto (upper left) and Lognormal (upper right) and actual traffic distribution (bottom).

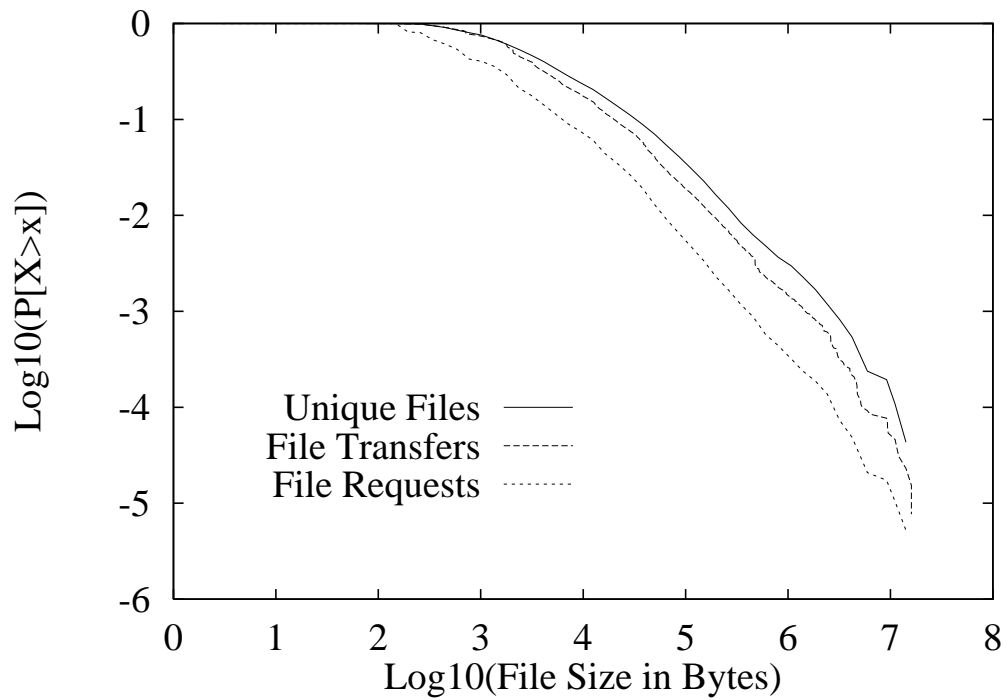
Heavy-tailed Transmission Times: Genesis

- Transmission times are heavy-tailed because the size distribution of transmitted files is heavy-tailed.
- The size distribution of transmitted files is heavy-tailed because the size distribution of unique requests is heavy-tailed.



Distributions of Requested, Unique, and Transferred Files

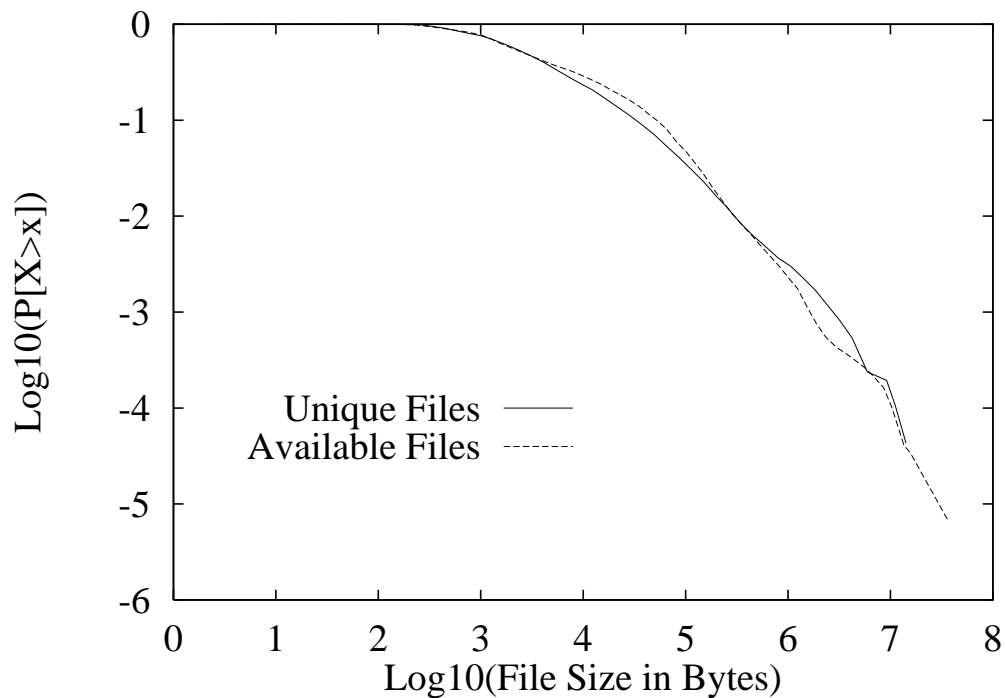
Distribution	α
Requested Files	1.16
Transferred Files	1.06
Unique Files	1.05



Why is the Size Distribution of Unique Files Heavy-Tailed?

- The size distribution of unique requests is heavy-tailed because it reflects the size distribution of available files on the Web, which we have shown to be heavy-tailed by surveying Web file systems at 32 different sites.

Distribution	α
Unique Files	1.05
Available Files	1.06



Relationship Between Various File Sets

Transmission
Times



File
Transfers



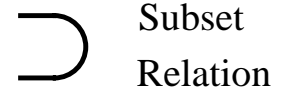
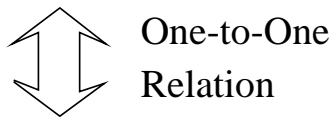
Unique
Files



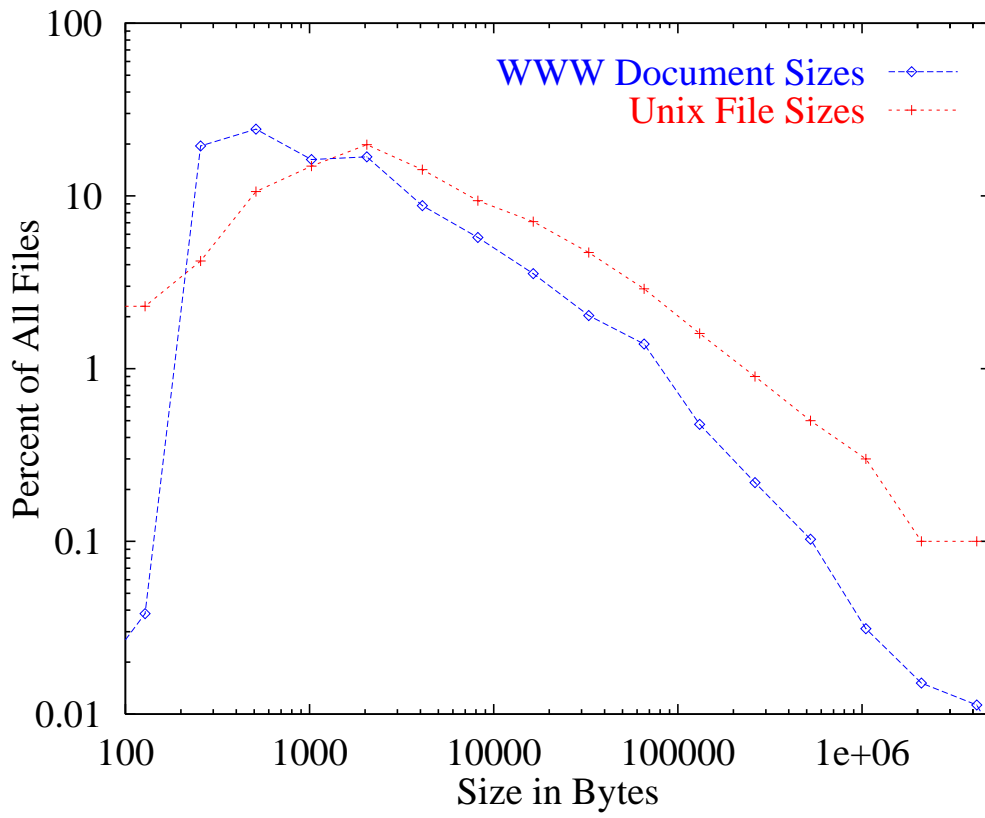
Available
Files



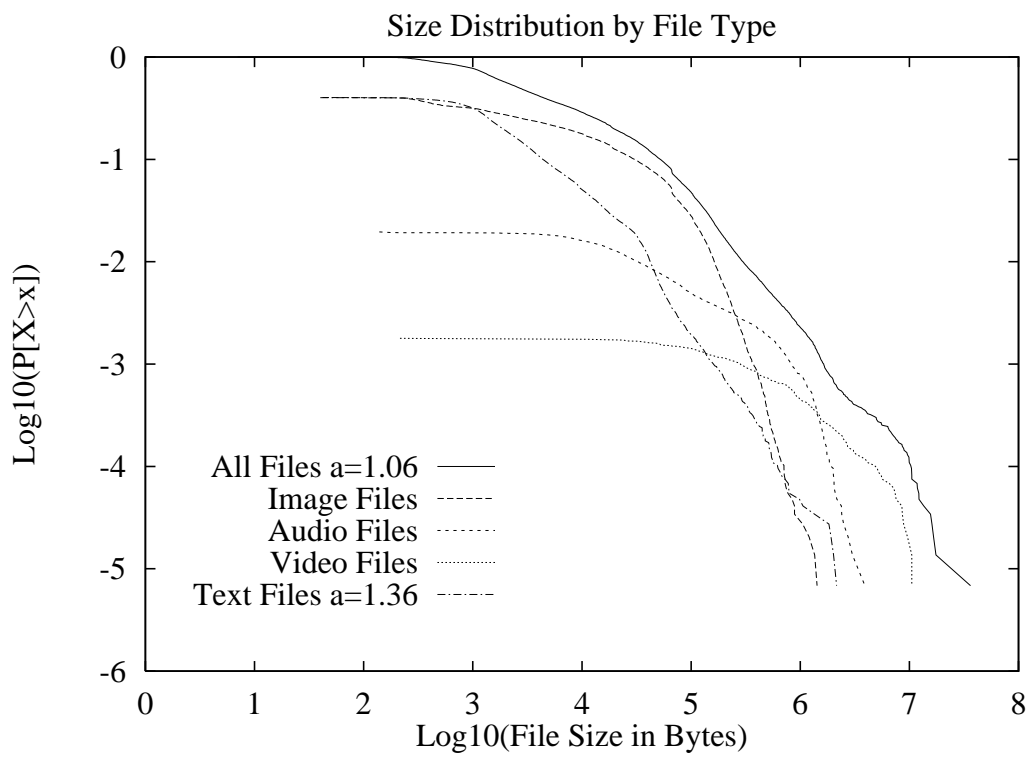
File
Requests



How Typical Are Web Files?



Relationship Between Heavy Tail and Media



Conclusion: Overview of OCEANS Projects

— Client-based Studies

- ◇ Client Caching Protocols
- ◇ Prefetching Protocols
- ◇ Dynamic Server Selection Protocols

— Server-based Studies

- ◇ Server Caching Protocols
- ◇ Dissemination Protocols
- ◇ Speculative Service Protocols

— Performance Modeling Studies

- ◇ Web Traffic Characteristics
- ◇ Web Access Patterns Characteristics

<http://cs-www.bu.edu/groups/oceans>