WebCompanion: A Friendly Client-Side Web Prefetching Agent

Reinhard P. Klemm

Bell Laboratory, Lucent Technologies
(Distributed Software Research Department)

IEEE TKDE, July/August 1999

Outline

- **Introduction**
- **WebCompanion Features**
- **Experimental Results**
- **Summary**

Introduction

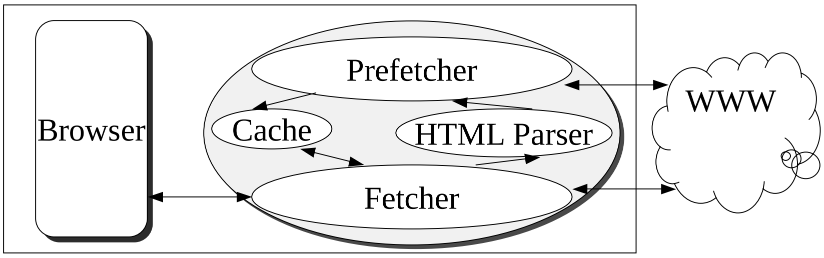
- **Motive**
  - Deal with the Web latency and bandwidth issues
  - Web prefetching strategies
    - ✓**client-side, proxy-side, server-side, hybrid**
  - Greedy prefetching strategy
    - ✓**high network overheads and resource consumption**
- **Goal**
  - Reduce the round trip time of accesses to the Web
  - Client-side Java-implemented prefetching agent
    - ✓**estimate the round trip times, limit the overhead**

Introduction

- **Main Idea**
  - Estimate the round trip times of all documents referenced by embedded hyperlinks
  - Prefetch the documents with the longest times first
- **Estimated Round-Trip Time-based Prefetching**
  - Highly selective prefetching strategy
    - ✓**only long retrieval latencies and low resource usage**
  - Sophisticated session control scheme
    - ✓**adapt to changing network and server conditions**
  - Startup prefetching, DNS caching

Introduction

- **Performance Gains**
  - Average speed-up > 50%
  - Average network byte overhead < 150%
- **Architecture**
  - HTTP Web proxy



WebCompanion Features

- **Server Statistics Cache**
  - Linearly weighted averages for individual servers
    - ✓**setup time: t_s**
    - ✓**waiting time: t_w**
    - ✓**byte transmission time: t_b**
    - ✓**resource size: s_r**
    - ✓**round trip time: $t_r = t_s + t_w + t_b * s_r$**

$$a_n = \frac{1 \times y_1}{n(n+1)/2} + \frac{2 \times y_2}{n(n+1)/2} + \ldots + \frac{n \times y_n}{n(n+1)/2}$$

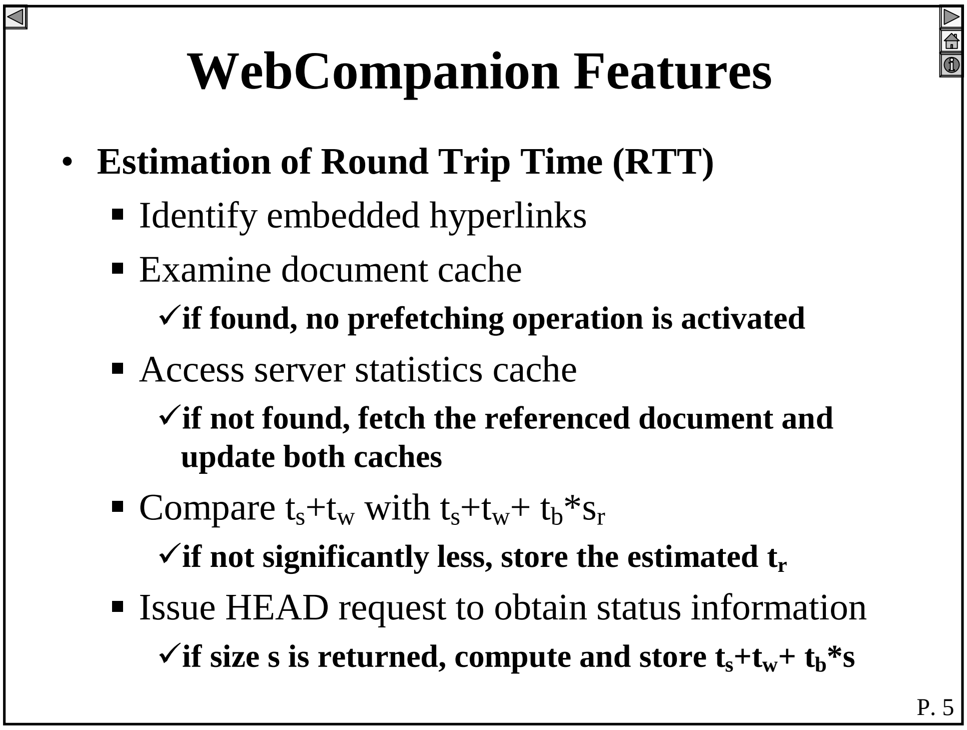
$$a_n = \frac{n-1}{n+1} \times a_{n-1} + \frac{2}{n+1} \times y_n$$

$$a_n = \omega \times a_{n-1} + (1 - \omega) \times y_n$$

1 2 7 1 2
a_2=1.67
a_3=4.33
a_4=3
a_5=2.67

1 2 1 2 7
a_2=1.67
a_3=1.33
a_4=1.6
a_5=3.4

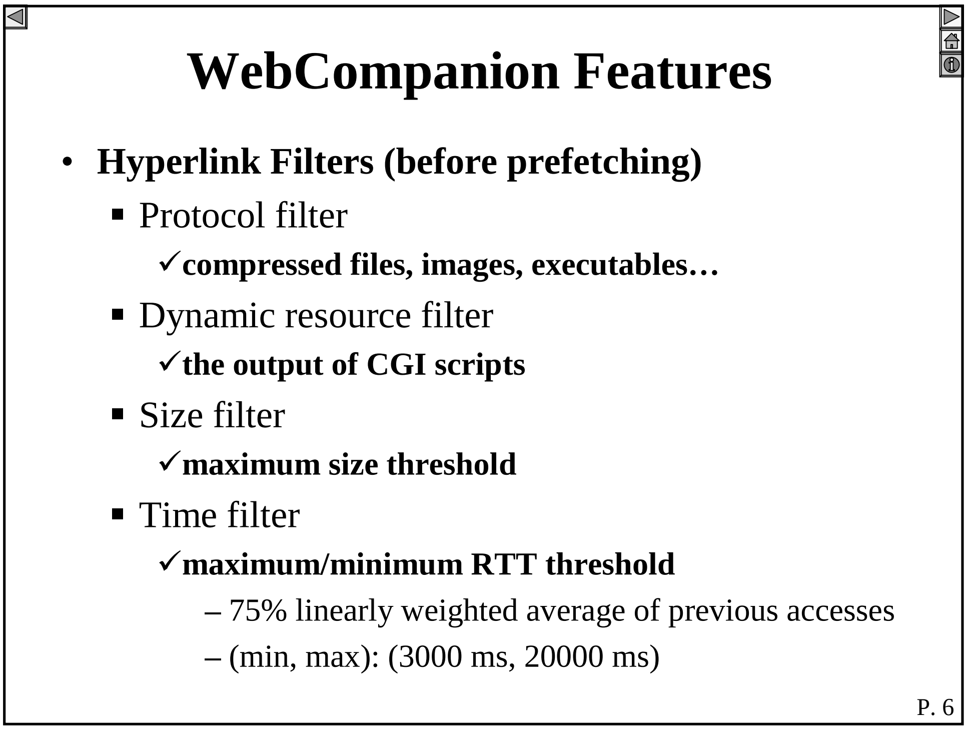
1 2 7 1 2
a_2=1.9
a_3=6.49
a_4=1.549
a_5=1.9549



WebCompanion Features

- **Estimation of Round Trip Time (RTT)**
  - Identify embedded hyperlinks
  - Examine document cache
    - ✓**if found, no prefetching operation is activated**
  - Access server statistics cache
    - ✓**if not found, fetch the referenced document and update both caches**
  - Compare t_s+t_w with t_s+t_w+ t_b*s_r
    - ✓**if not significantly less, store the estimated t_r**
  - Issue HEAD request to obtain status information
    - ✓**if size s is returned, compute and store t_s+t_w+ t_b*s**

P. 5



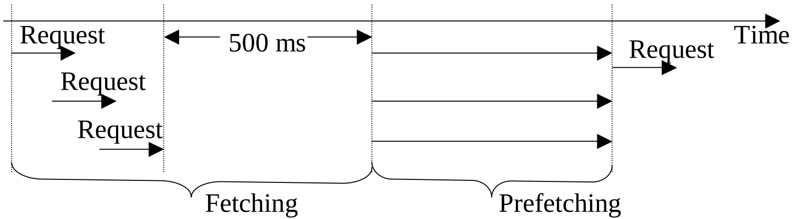
WebCompanion Features

- **Hyperlink Filters (before prefetching)**
  - Protocol filter
    - ✓**compressed files, images, executables…**
  - Dynamic resource filter
    - ✓**the output of CGI scripts**
  - Size filter
    - ✓**maximum size threshold**
  - Time filter
    - ✓**maximum/minimum RTT threshold**
      - – 75% linearly weighted average of previous accesses
      - – (min, max): (3000 ms, 20000 ms)

P. 6

WebCompanion Features

- **Session Control**
  - Start prefetching
    - ✓**first request arrives ⇒ the beginning a new session**
    - ✓**new request arrives ⇒ examine the ongoing session**
  - Stop prefetching
    - ✓**restart a new session with a short delay (500 ms)**
      - – wait for requests to reconsider the end of session



WebCompanion Features

- **Implementation**
  - Memory cache with compression
    - ✓**2MB cache with LRU replacement policy**
    - ✓**65.44% cache hit ratio**
      - – RTT vs. priority to be displaced
  - DNS caching
    - ✓**influence on the pessimistic scenario**
  - Parallel prefetching
  - GUI on statistics and cache

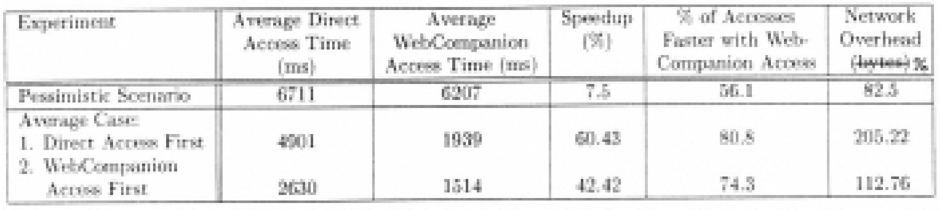
Experimental Results

- **WebWatch**
  - Workload generator (browser emulator)
    - ✓**URL list with a configure probability distribution**
      - – no duplicate requests/no cache
    - ✓**switch between WebCompanion and direct access**
  - Performance measurement
    - ✓**pessimistic access pattern**
      - – simulate user idle time: fixed interval (5 sec)
    - ✓**average-case access pattern**
      - – simulate user idle time: Poisson distribution (λ=10)
      - – randomly select hyperlinks: equal probability (80%)

Experimental Results

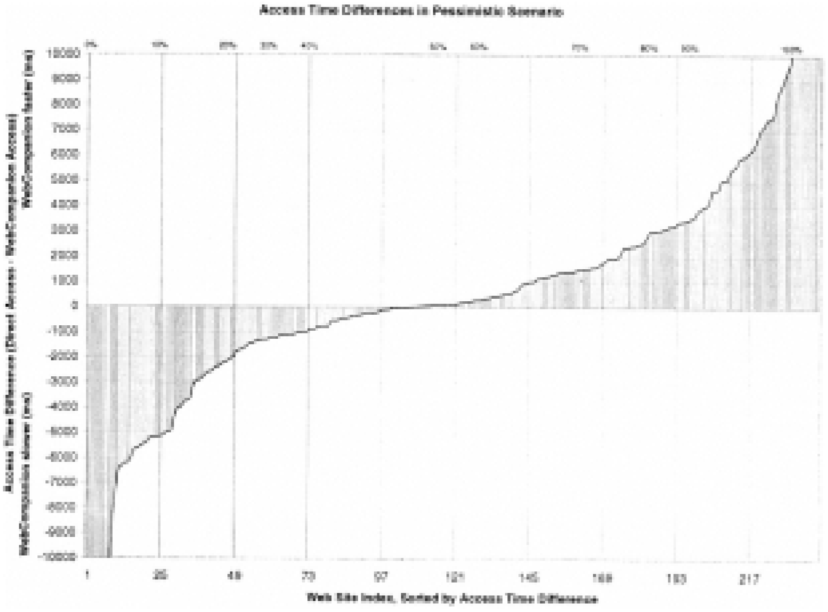
- **Comparison**
  - Speedup
  - Percentage of faster accesses
  - Network overhead

Comparison Between Direct Access and WebCompanion Access; Averages Are Per Web Resource



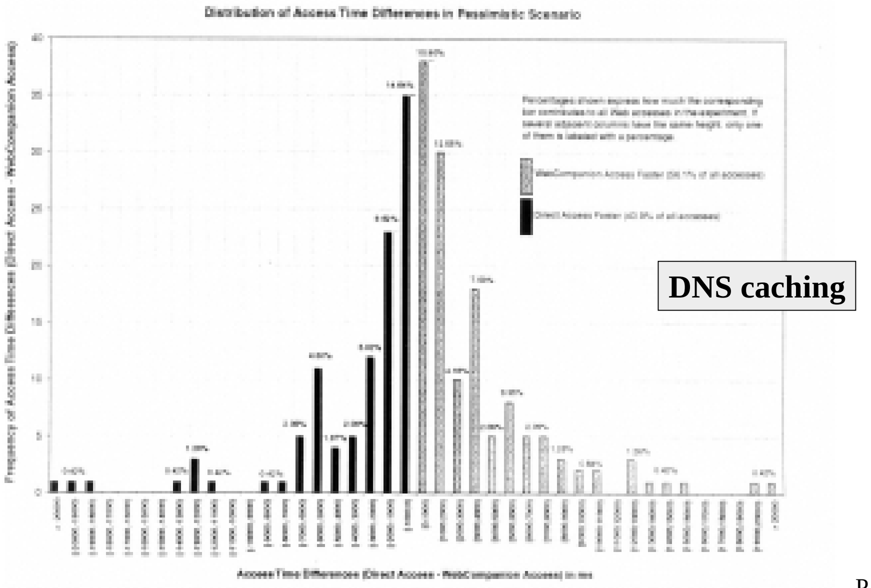
Experiment	Average Direct Access Time (ms)	Average WebCompanion Access Time (ms)	Speedup (%)	% of Accesses Faster with Web-Companion Access	Network Overhead ~~(bytes)~~ %
Pessimistic Scenario	6711	6207	7.5	56.1	82.5
Average Case: 1. Direct Access First	4901	1939	60.43	80.8	205.22
2. WebCompanion Access First	2630	1514	42.42	74.3	112.76

Experimental Results



P. 11

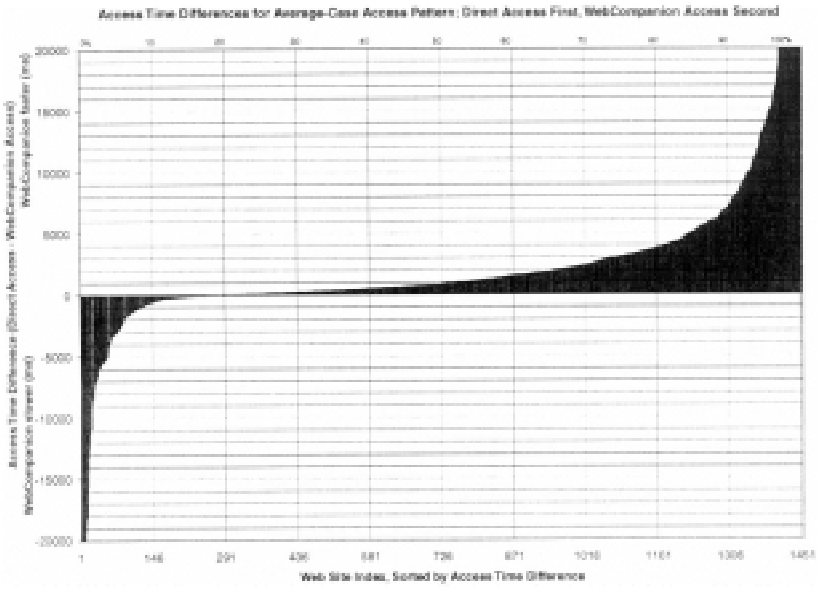
Experimental Results



DNS caching

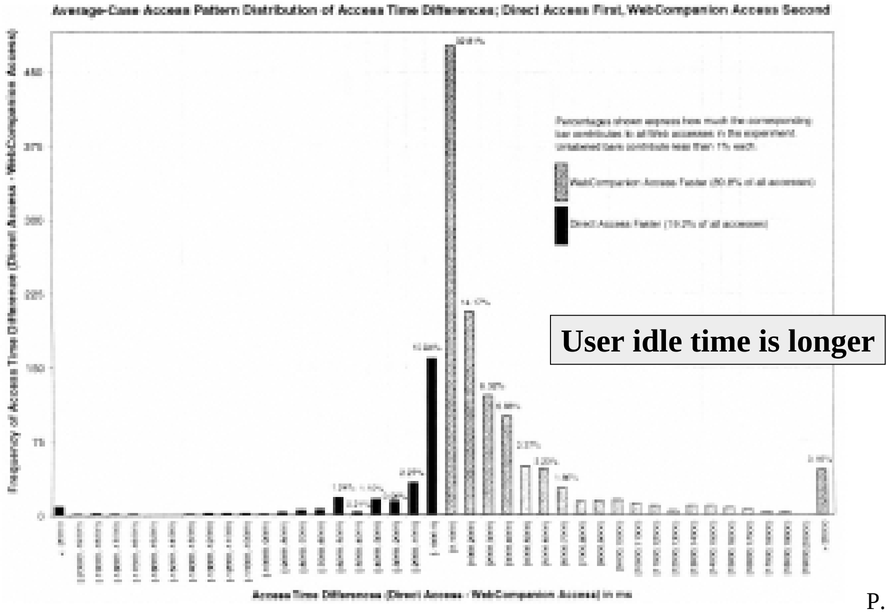
P. 12

Experimental Results



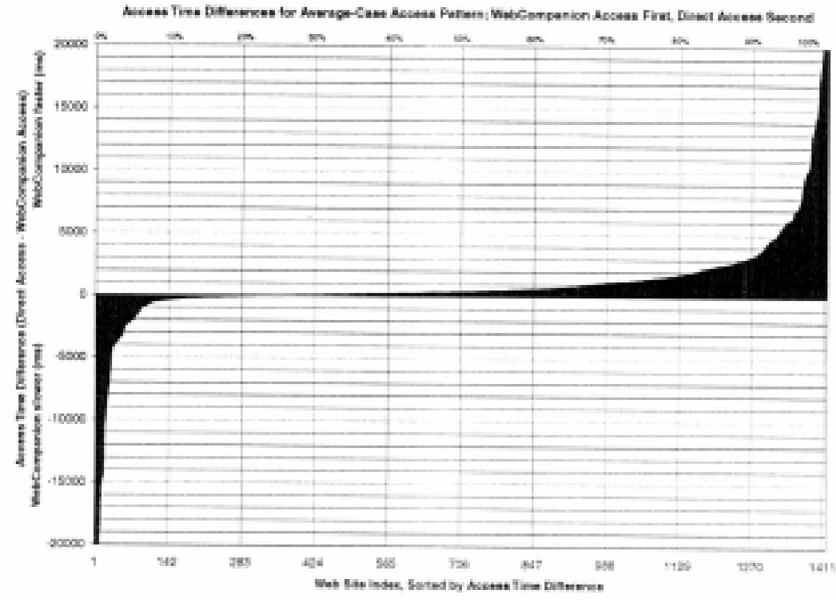
l

Experimental Results



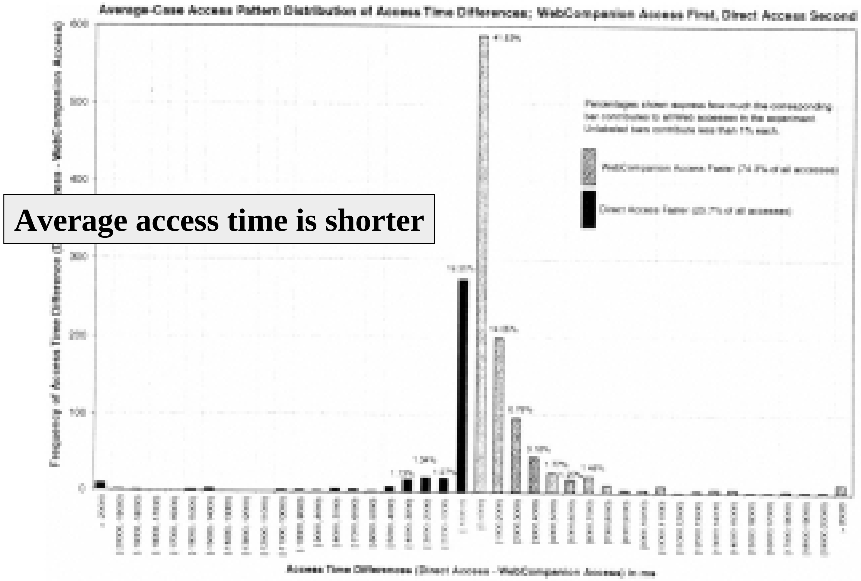
User idle time is longer

Experimental Results



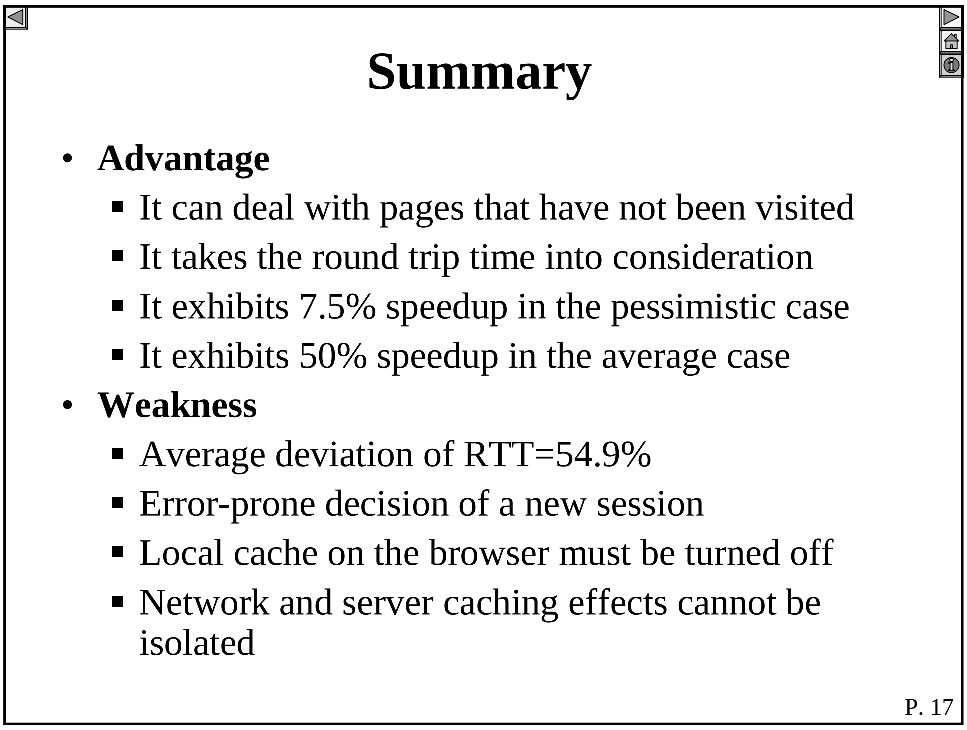
P. 15

Experimental Results



Average access time is shorter

P. 16



Summary

- **Advantage**
  - It can deal with pages that have not been visited
  - It takes the round trip time into consideration
  - It exhibits 7.5% speedup in the pessimistic case
  - It exhibits 50% speedup in the average case
- **Weakness**
  - Average deviation of RTT=54.9%
  - Error-prone decision of a new session
  - Local cache on the browser must be turned off
  - Network and server caching effects cannot be isolated