Latency Aware Information Access with User Directed Caching

Web *Vade Mecum*

James P.G. Sterbenz*

jpgs@acm.org

+1 508 944 3067

Tushar Saxena

Rajesh Krishnan

{tsaxena|krash}@bbn.com

*Work originally performed while at BBN technologies, supported by DARPA contract N66001-97-D-8622

Abstract

Mobile wireless clients such as a laptops and PDAs, particularly on ships or airplanes, have highly variable network connectivity and available bandwidth. There are times when they have poor connectivity to the Internet, or may be completely disconnected. This poses unique constraints for the problem of information access in general, and web access in particular.

We are concerned with three related problems:

1. Keeping the cache as current as practical using techniques such as demand prefetching and push preloading. During periods of poor bandwidth or episodic disconnectivity this requires that the client, proxy, and caching infrastructure be aware of network traffic conditions.

2. When desired content is either not fresh or absent, the behaviour of the client should be based on user desire, for example, whether to wait for the content, use an old cached version, or both. This decision should be based in part on an estimate provided to the user on the estimated latency to fetch the content, and can be made based on a profile of past history or an explicit user interaction loop.

3. During periods of poor connectivity and low bandwidth it is important to schedule and prioritise current requests against preloading/prefetching to optimise future access.

This talk will present the problem, motivation, architecture, prototype implementation in progress, and research issues to be explored. Our work draws on the experiences of designing mobile clients in other scenarios such as distributed file systems (eg. Coda), as well the large body of work on Web caching and anticipation.

Outline

- Motivation, Environment, and Background
  - application
  - environment
  - caching and anticipation
- Problem and Proposed Solution
  - latency aware client
  - user emulation
  - user directed handling of misses
- Prototype
  - operation and information flow
  - profile creation and walking
  - user interface to latency information
- Further research issues
  - accurate estimation of retrieval time
  - request scheduling

Motivation and Environment

- Motivation, Environment, and Background
  - application
  - environment
  - caching and anticipation
- Problem and Proposed Solution
- Prototype
- Further research issues

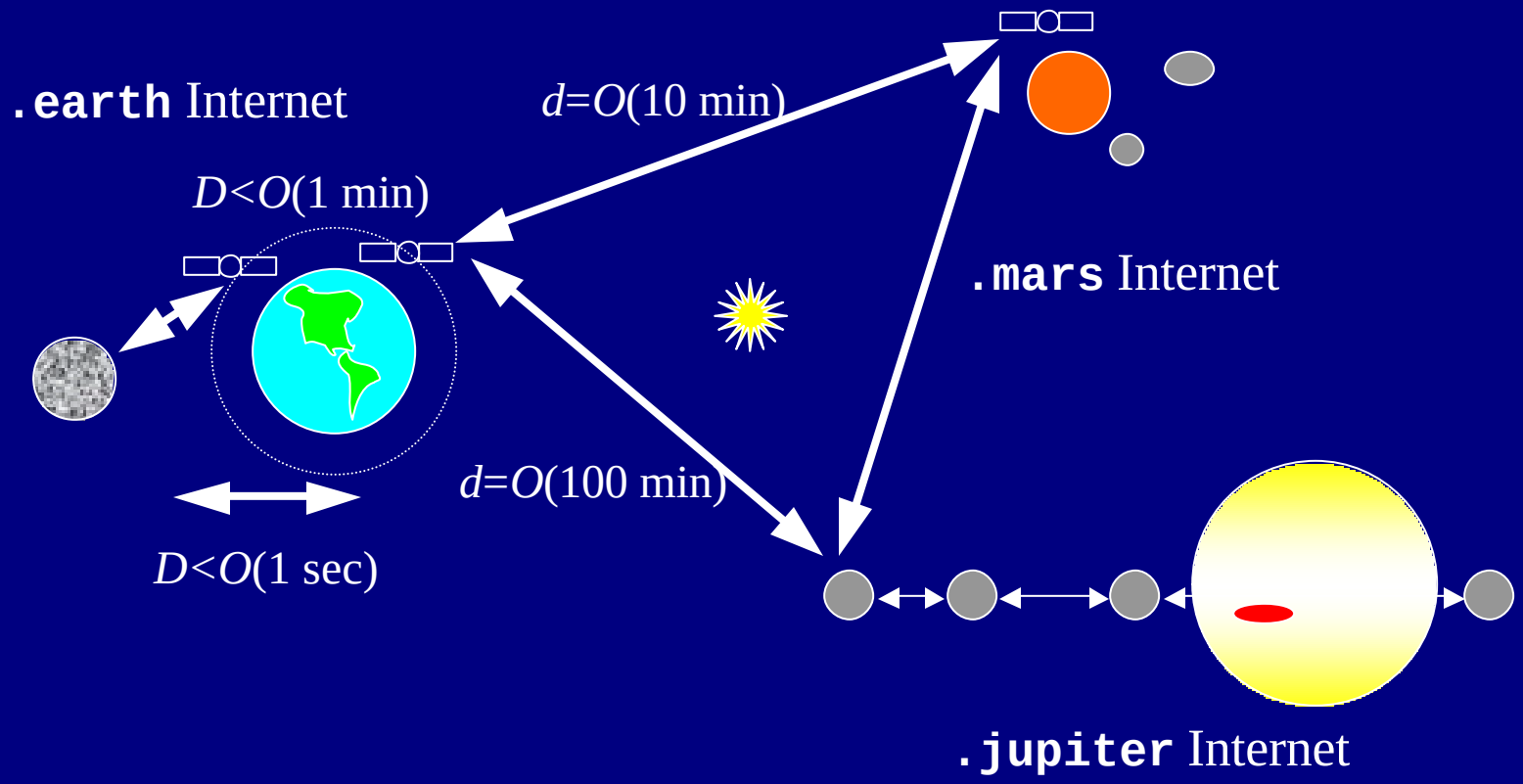
Motivation
Application

- Distributed information access
  - access information from remote locations
  - Web provides most common infrastructure
    - web browser as client
    - HTTP as protocol
- User experience highly dependent on response time
  - *interactive* information access
    - subsecond target response time
    - 100 ms ideal response time

Motivation
Application

- Applications should be usable even when
  - long latency
    - long links
    - slow servers
  - weakly connected
  - disconnected (episodic)

Environment
Long Latency Links



Environment
Mobile Wireless Links

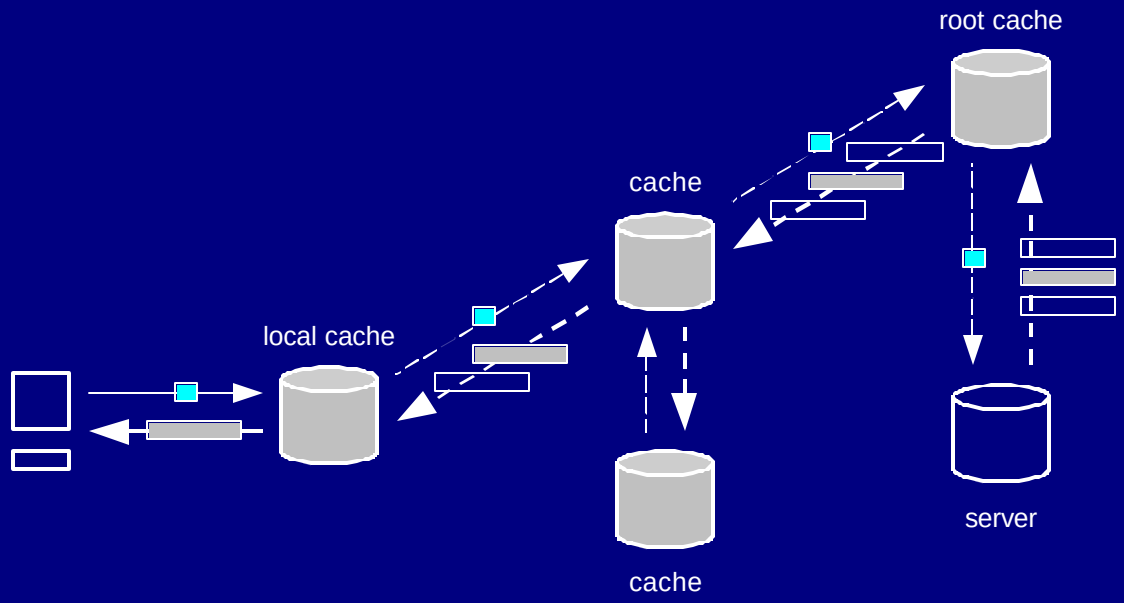
- Weak connectivity with limited bandwidth
  - contention for shared medium
  - highly variable channel conditions
- Episodic connectivity
  - long channel fades (rain, occultation)
  - long periods of disconnectivity (caves, faraday cages)
  - jammed channels (noise or bad guys)
  - radio silence (bad guys near)
- Congested networks
  - wireless or wired

Background
Masking the Speed of Light

Traditional techniques to mask speed of light latency

- Caching
  - on-demand fetching
  - maintain working set of recently used pages
  - expect that pages will be requested again
  - difficult for dynamic content
- Anticipation

Caching
Hierarchy

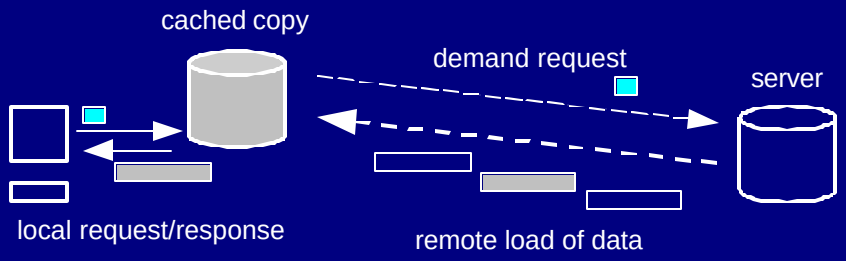


- Cache frequently organised as tree (e.g. squid)
  - cache miss results in fetch attempt up the tree
  - root cache miss results in fetch to definitive server
  - exploits locality groups at multiple levels

Anticipation

- Caching
- Anticipation
  - prefetching
  - push preloading
  - datacycle

Anticipation
Prefetching



- Fetch pages likely to be referenced in future
  - fetch pages in hyperlink reference tree
    - tree depth based on latency and bandwidth available
  - fetch based on previous user behaviour

Anticipation
Push Preloading

- Server preloads pages based on expected behaviour
  - past behaviour
  - registration to application services (e.g. news)
- Reduces instantaneous bandwidth demand
  - trickle in advance vs. burst on demand

Problem and Proposed Solution

- Motivation, Environment, and Background
- Problem and Proposed Solution
  - latency aware client application
  - user emulation
  - user directed handling of misses
- Prototype
- Further research issues

Problem

- Caching and anticipation
  - maximise probability of cache hit
  - *but content may not be cached (or fresh)*
  - no scheduling for weak and episodic connectivity
- Penalty of cache miss may be very high
  - very long latency links
  - poor quality links
  - episodic connectivity
- User has no control over resolution of miss
  - time out to HTTP 404 message not desired result

Proposed Solution

- Latency *aware* applications
  - transparency hides too much; translucency better
  - traditional caching and anticipation as much as possible
  - client presents response time estimates to user
  - emulates user behaviour when
    - user not present
    - network strongly connected
- *User directed* handling of cache misses
  - explicit user direction on fetch behaviour
  - profile based on past preferences

Proposed Solution
Latency Aware Applications

- Application should be aware of
  - freshness of cached information
  - latency to retrieve definitive copy of information
- Requires knowledge of
  - freshness metadata
  - size of information object (first usable increment)
  - channel bandwidth
  - channel RTT
  - time to reconnect if disconnected
    - perhaps mission based
    - based on past behaviour

Proposed Solution
Fetch Based on Past Behaviour

- Build profile graph of user behaviour
  - nodes are pages
  - links are transitions
- Emulate user when
  - user not actively accessing information and
  - strongly connected to network
  - be prepared for disconnected operation
- Schedule fetches based on past behaviour
  - e.g. fetch CNN weekday mornings before commute to work

Proposed Solution
User Directed Handling of Cache Misses

- User influences behaviour when not in cache or old
  - use older cached copy
  - wait for newer/definitive copy } or both
    - spin off to background; notify when fetched
  - do something completely different
- User interaction
  - explicit interaction: right-click options
  - profile-based
    - explicit defaults
    - automatic learning

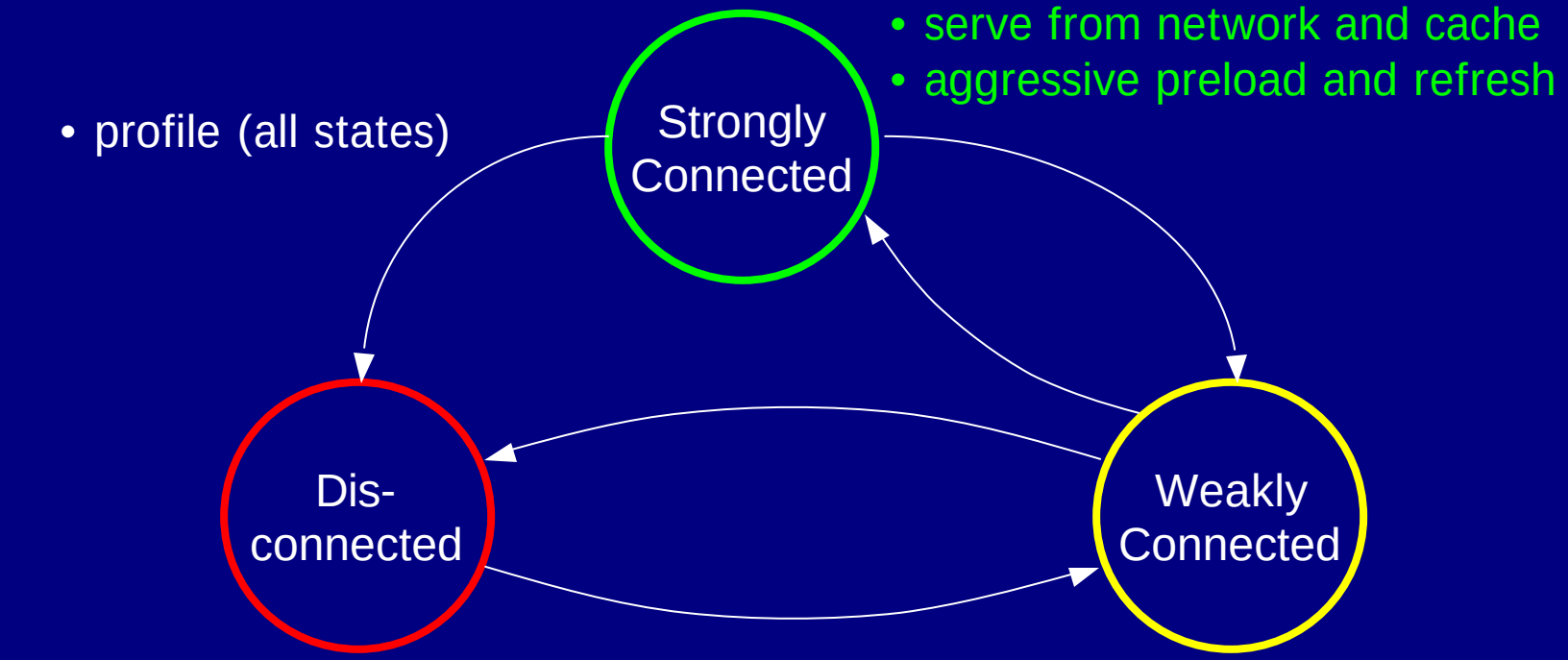
Proposed Solution
Scheduling of Preloading and Refreshing

- Bandwidth demand exceeds link capacity
  - weak and episodic connectivity
  - multiple users and applications
  - aggressive preloading and refreshing
- Schedule among:
  - current requests
  - cache refreshes
  - anticipatory prefetch and push preload
- Based on:
  - value of information
  - network traffic conditions

Prototype

- Motivation, Environment, and Background
- Problem and Proposed Solution
- Prototype
  - operation and information flow
  - profile creation and walking
  - user interface to latency information
- Further research issues

Prototype Operation
Proxy States [CODA]



Prototype Operation
Strongly Connected Mode

- Pass the request directly to cache hierarchy
  - give user estimate to retrieve and options if long latency link
- Monitor and profile user requests
- Aggressively refresh cache: hoarding
  - conventional anticipation
  - inform server to push preload

Prototype Operation
Disconnected and Weakly-Connected Mode

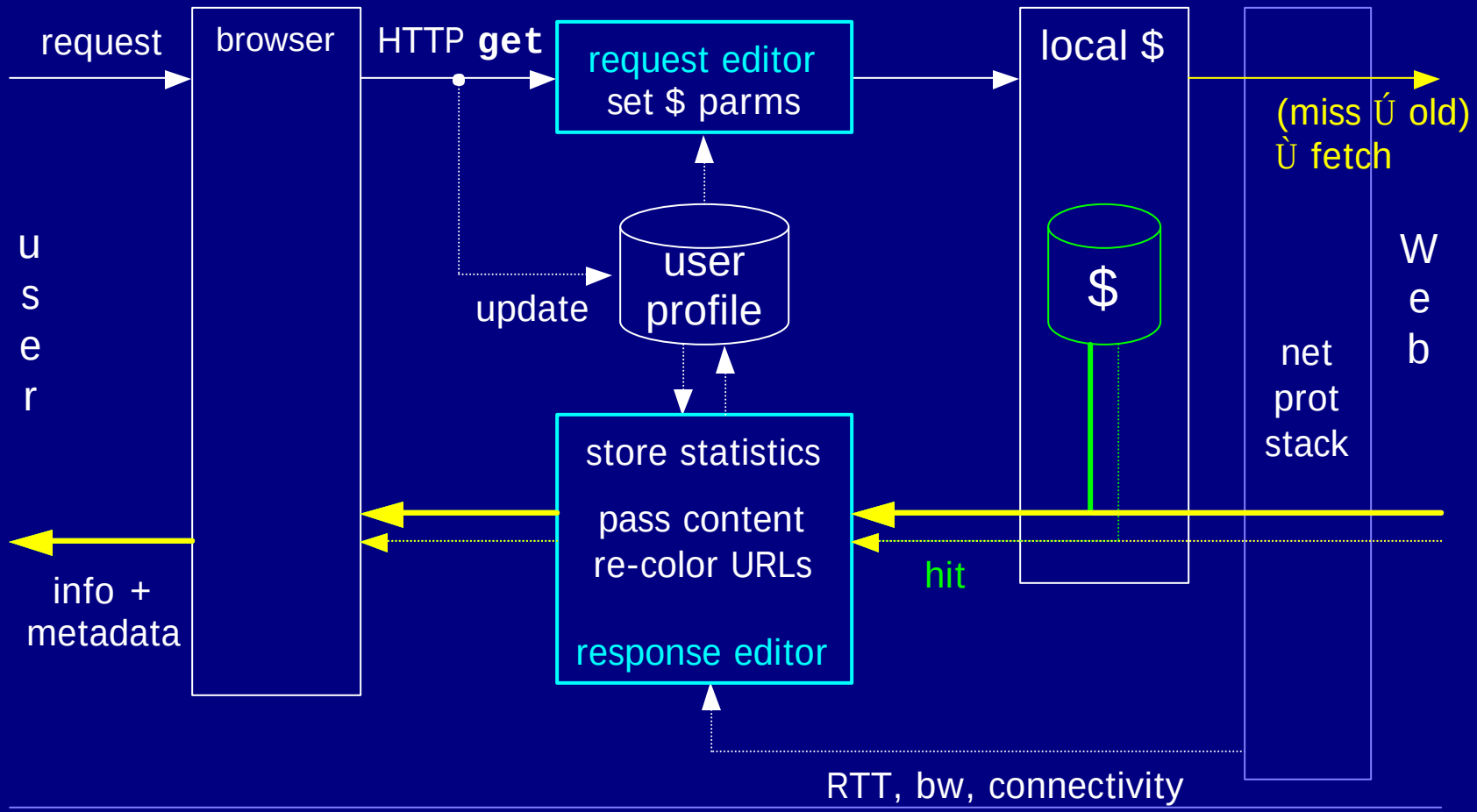
- Serve previously cached pages
  - provide freshness estimate based on timestamps/version
- Estimate time for fetching fresher pages
  - use profile defaults when present
  - seek user advice for fetching page and updating cache
  - inform user of reconnection estimates
- If weakly connected: trickle update cache
  - use available bandwidth
  - highest priority pages
  - seek user advice on future requests

Prototype Implementation

- Browser – *Mozilla*
  - menu item modifications (view, right click)
- Client proxy request/response editors – IBM *WBI*
  - request editor: snoop on requests to build profile
  - response editor: rewrite HTML to add latency information
  - interaction with network characteristics
- Cache proxy – *squid*
  - minimal modifications to interface with client proxy

Platform for research into interesting issues

Prototype Operation
Information Flow



User Profiles

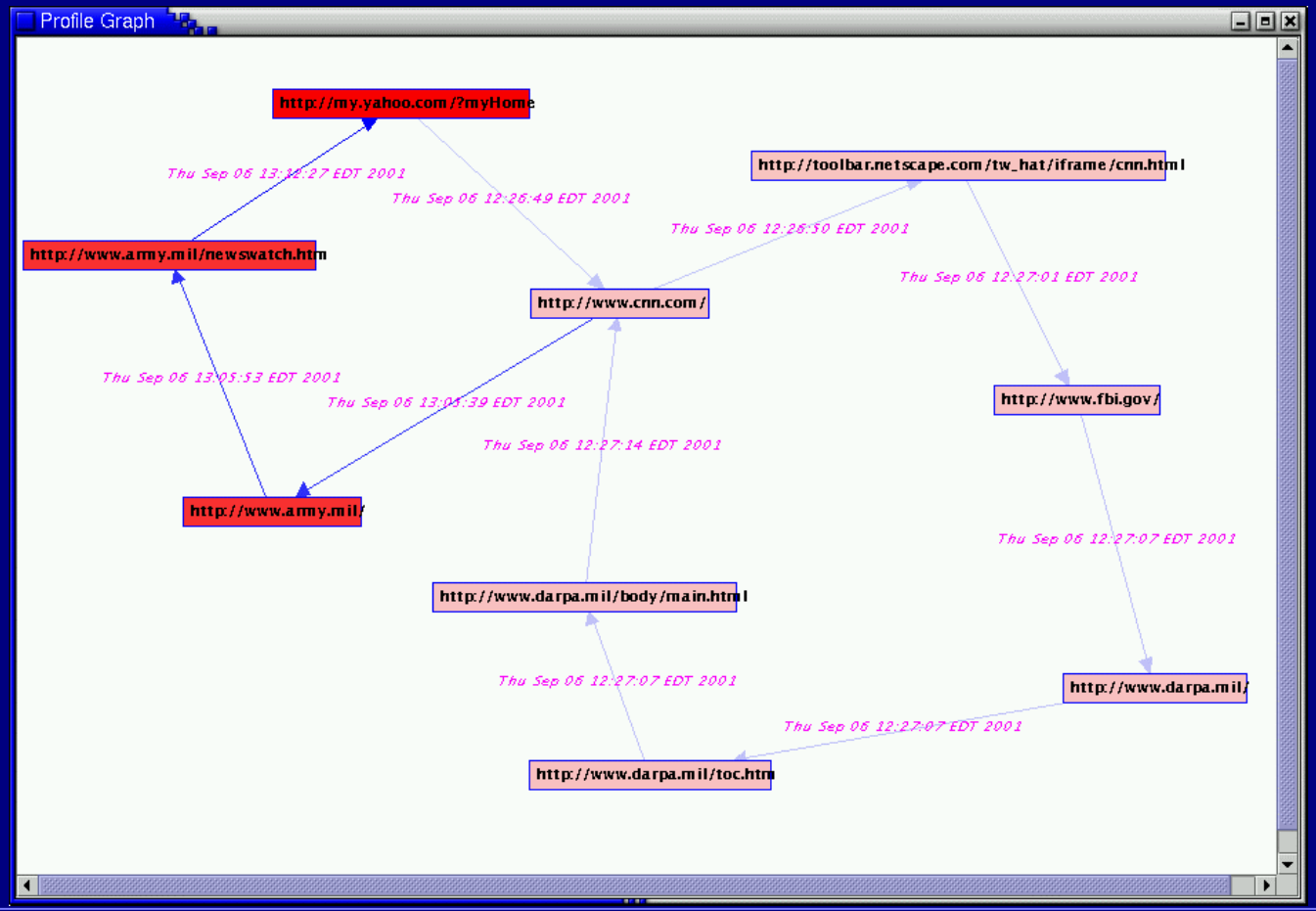
- All user requests pass through WVM
  - in all modes: fully, weakly, and disconnected
- WVM builds a user-specific *WVM-ProfileGraph*

Profile Graph

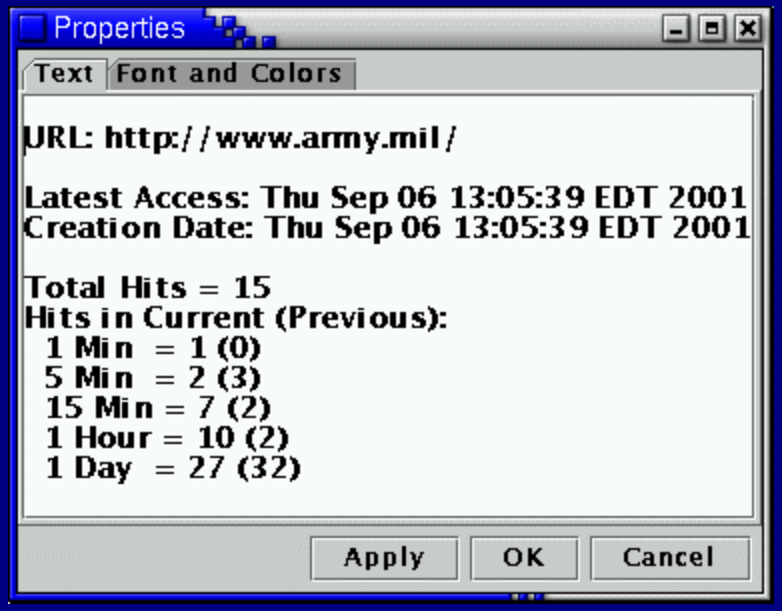
- Tracks and represents user Web access pattern
  - nodes are URLs accessed
  - directed edges indicate URLs traversed
- URL priority limits cache life and refresh bandwidth
  - nodes and edges fade using a *decay Formula*
  - other mechanisms for importance of pages to be explored
- Autonomic user emulation
  - graph is automatically traversed when fully connected
  - graph slowly/selectively traversed when weakly connected
  - cache kept fresh even when user away from client
  - cache is as fresh as possible after state change (weak|disc)

WVM Profile Graph

Example Screenshot

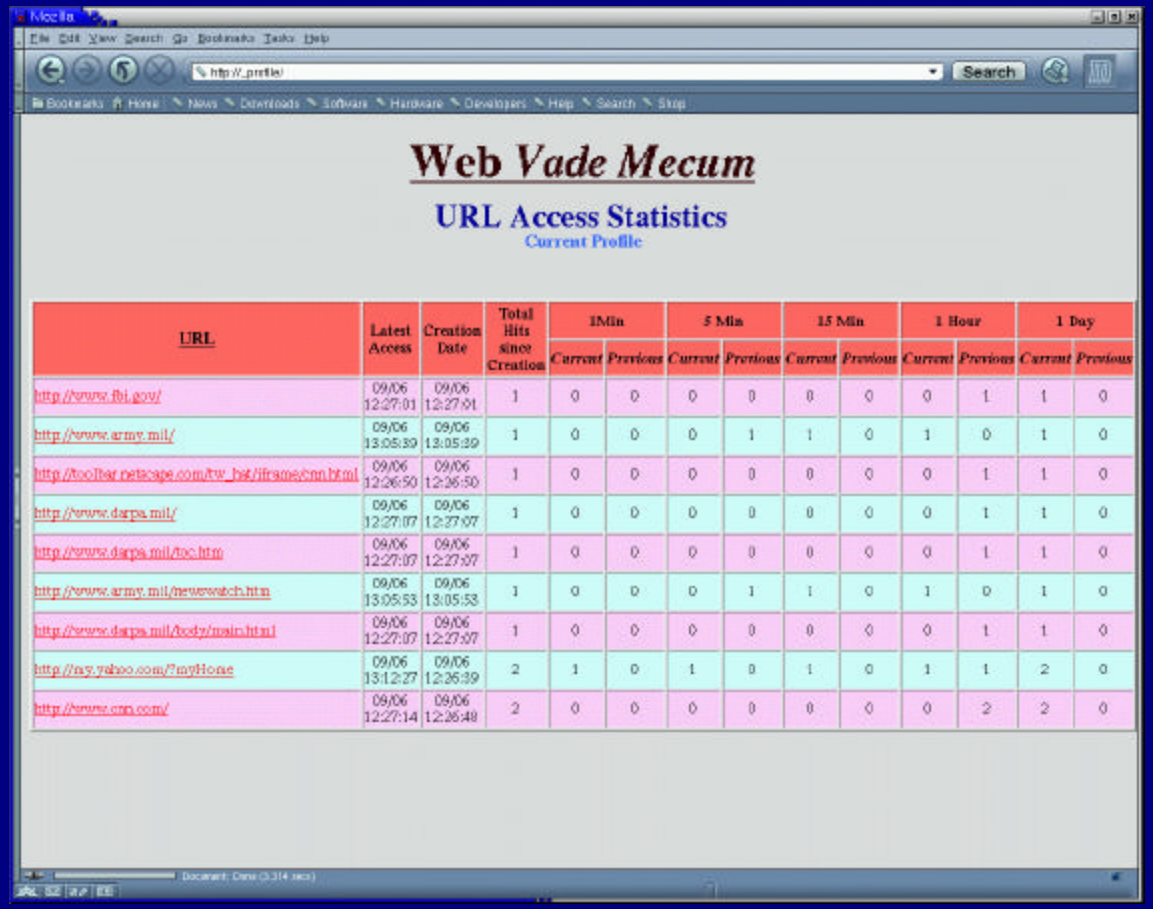


WVM Profile Graph
Individual URL Statistics



WVM Profile Graph

Profile Database



Web *Vade Mecum*

URL Access Statistics

Current Profile

URL	Latest Access	Creation Date	Total Hits since Creation	1Min		5 Min		15 Min		1 Hour		1 Day	
				Current	*Previous*	*Current*	*Previous*	*Current*	*Previous*	*Current*	*Previous*	*Current*	*Previous*
http://www.fbi.gov/	09/06 12:27:01	09/06 12:27:01	1	0	0	0	0	0	0	0	1	1	0
http://www.army.mil/	09/06 13:05:39	09/06 13:05:39	1	0	0	0	1	1	0	1	0	1	0
http://toolbar.netscape.com/tw_hit/iframe/cnn.html	09/06 12:26:50	09/06 12:26:50	1	0	0	0	0	0	0	0	1	1	0
http://www.darpa.mil/	09/06 12:27:07	09/06 12:27:07	1	0	0	0	0	0	0	0	1	1	0
http://www.darpa.mil/toc.htm	09/06 12:27:07	09/06 12:27:07	1	0	0	0	0	0	0	0	1	1	0
http://www.army.mil/newswatch.htm	09/06 13:05:53	09/06 13:05:53	1	0	0	0	1	1	0	1	0	1	0
http://www.darpa.mil/body/main.html	09/06 12:27:07	09/06 12:27:07	1	0	0	0	0	0	0	0	1	1	0
http://my.yahoo.com/?myHome	09/06 13:12:27	09/06 12:26:39	2	1	0	1	0	1	0	1	1	2	0
http://www.cnn.com/	09/06 12:27:14	09/06 12:26:48	2	0	0	0	0	0	0	0	2	2	0

Latency-Based User Interface

- Links colored according to access characteristics
  - *green* close or fresh cached $\hat{t}_r < t_i$ $\hat{a} < a_i$
  - *yellow* cached but not fresh $\hat{t}_r < t_i$ $\hat{a} > a_i$
  - *red* long latency to receive $\hat{t}_r > t_i$ $\hat{a} > a_i$
  - *blue* no reasonable estimate possible
- Status bar at bottom of browser window
  - connectivity information
  - per URL information on mouse-over
    - freshness information a
    - estimates on t_r

Note: goal is to explore two levels of information, not do human factors study

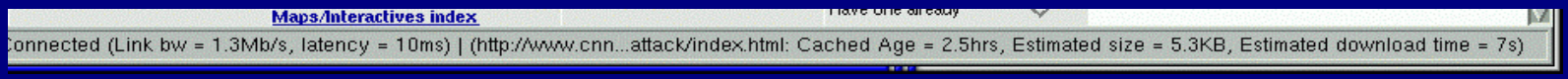
Connectivity Information

- Measured to *lighthouse* across known weak link
  - bandwidth
  - latency
- Sampled periodically to adjust response time est.
- Examples:
  - wireless LAN lighthouse at base station
  - ship or plane lighthouse at fixed access nodes

Latency-Based User Interface

Complex Page with Images

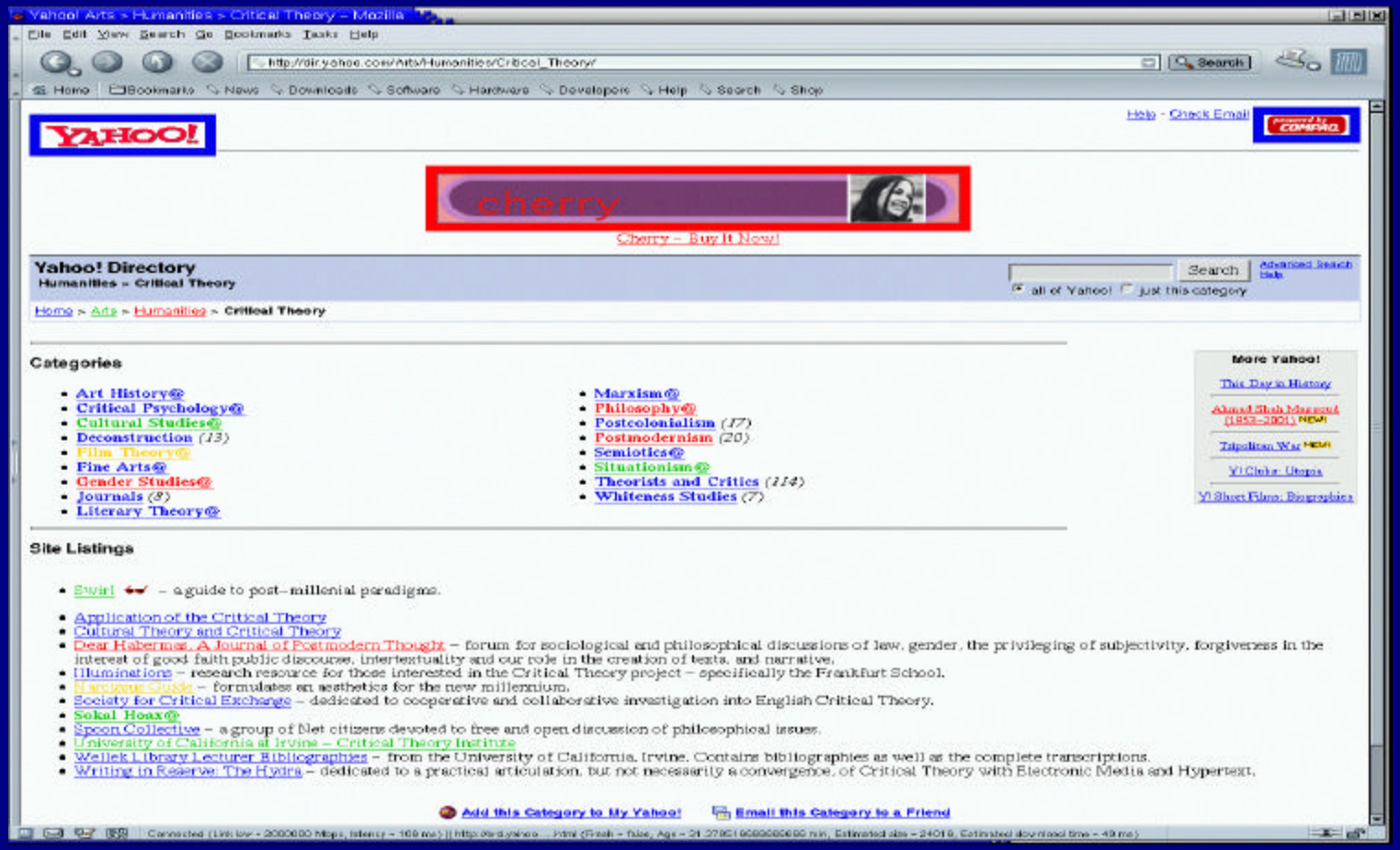
- Link color gives high level {*fast*, *old*, *slow*, *unknown*}



- Status bar indicates global and *per* link details

Latency-Based User Interface

List of URLs



User Interaction Loop

- Click type determines fetch action
  - left click: “normal” behavior
    - get cached if available
    - profile based action
  - right click gives options
    - fetch definitive

      refresh window when definitive copy arrives
    - nonblocking fetch

      definitive copy in new window when available
- View menu selection
  - allows display of unmodified page

Dynamic Content

- Too much dynamic content on the Web
  - overuse of gratuitous dynamic content (e.g. decoration)
  - dynamically generated URLs also problematic
  - should be optimised for weakly and disconnected operation
- This project doesn't attempt to solve this, but...
  - static *and* locally generatable content should be cacheable
    - organise data into cacheable units
  - applet vs. servlet tradeoff
    - applets are cacheable / prefetchable

Further Research Issues

- Motivation, Environment, and Background
- Problem and Proposed Solution
- Prototype
- Further research issues
  - accurate estimation of retrieval time
  - request scheduling

Issues
Estimation of Retrieval Time

- Network characteristics
  - end-to-end latency
    - history
    - periodic pings to anticipated remote servers
- Object size
  - history for cached objects
  - metadata in higher level pages
- Currently use last access and lighthouse

How accurate is possible and necessary?

Issues
Request Scheduling

- Schedule get requests among:
  - current user request
  - pending user requests
  - cache refreshing
  - cache preloading (inform server whether to push preload)
- Optimise
  - long term per user
    - current request not necessarily highest priority
  - over multiple users
    - e.g. on board ship or plane
  - based on expected weak/disconnectivity episodes

Acknowledgements

This work funded by DARPA ITO contract

- Jean Scholtz, PM

In collaboration with

- Mike Sullivan, BBN

Feedback from presentations and discussions:

- Fraunhofer Instutut FOKUS
- IBM Research, Zürich
- Technical University of Saarbrücken, Germany
- ETH Zürich
- USC/ISI
- Technical University of Karlsruhe, Germany
- University of Bern, Switzerland
- UCI (University of California at Irvine)
- University of Kentucky

End Of Foils