Cut-Load: Application-Unaware Content Partitioning for Web-based Information Access in Wireless Data Networks

Tae-Young Chang,
Aravind Velayutham,
and Raghupathy Sivakumar

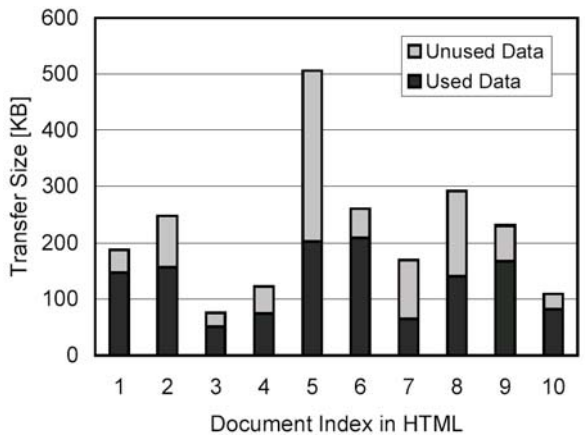
School of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, GA 30318

Introduction

- *Mobile Information Access (MIA)*
  - All the work performed by the mobile host as non-local information access
    - ex) web-browsing, remote file access, e-mail, etc.
  - The ongoing migration of information access from the desktop to mobile computing devices poses critical challenges for research in information access.
    - The conventional information access models lead to the problems of excessive bandwidth consumption and large response delays in mobile environments.
- In this work, (1) we analyze the reason for inefficient information access in mobile environments and (2) propose a new paradigm for mobile information access that is driven by user-activity awareness.

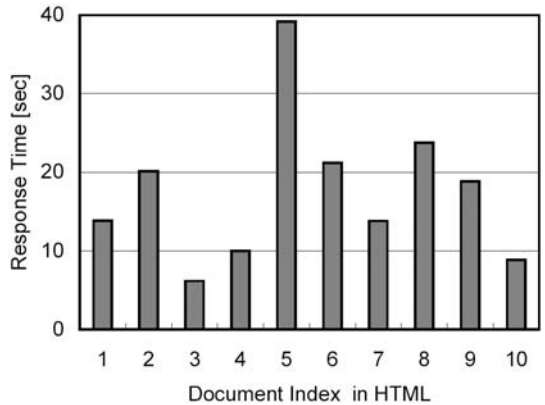
Motivation (1)

- Not all content is always seen by users.
  - Users generally don't view the entire content of a fetched file, and may see only partial content that they are interested in.
    - 90% of users don’t scroll down web pages but simply pick from the options that is visible on the initial screen when a page comes up. [NIELSEN99]
  - However, the conventional computing always consumes additional bandwidth to fetch unnecessary parts of content
    - The figure shows the total data transfer size and the amount of data that the user ends up using for the 10 web document files.
    - A significant difference exists in transfer sizes between full data transfer and the necessary content.



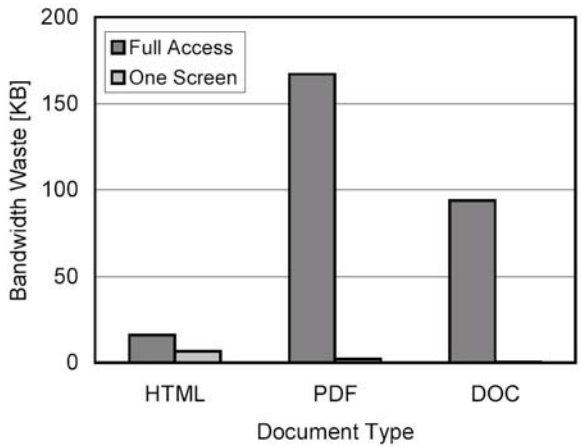
Motivation (2)

- Users suffer when response times are large.
  - Relationship between response time and users' perceptions
    - Users lose their concentration on their access when the response time is larger than 10 sec. When it is over 1 minute, they lose interest and stop the current access [MILLER68].
  - A mobile client waits until the most content of a document is fetched regardless of which part a user wants to see.
    - Fetching the unnecessary part of content increases the initial response time significantly and makes users impatient.
    - The figure shows the response time results for the same set of the documents.
    - Download of only useful content can minimize response time without the degradation of content quality.



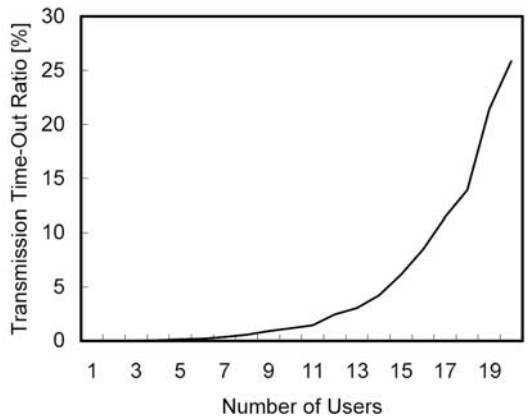
Motivation (3)

- Larger file transmissions suffer from frequent disconnections.
  - Wireless networks are prone to frequent disconnections.
    - Signal attenuation, channel fading, interference, mobility, etc.
  - The partially downloaded file until disconnection is generally not usable for serving the user requests for content.
    - The bandwidth expended to download the file goes waste.
    - The response time perceived by the user is increased because of the additional time spent in downloading the content again.
    - The figure shows that the amount of bandwidth wastage is significantly smaller in the ideal case
    - In the ideal case, the probability of transmission failures caused by disconnections is smaller due to its smaller transfer time.



Motivation (4)

- Greedy transmission makes network utilization inefficient.
  - We say that a connection has timed-out if the response time for the connection is greater than the specified latency tolerated by the user, and the user has stalled the download.
  - When the connection is reset by the user, the bandwidth used in downloading the content till that point is wasted.
    - The figure shows the percentage of connections that timeout as a function of the number of users in the network.
    - The time-out rates increase exponentially when the number of users (network load) is linearly increased.
    - The higher peak load on the system degrades the system utilization and hence decreases the performance of the connections.



Challenge and Goal

- *User-activity unawareness*
  - Current access models are not able to differentiate between the essential part of the file and the other part that will not be used by the application.
    - A content file requested by an application is retrieved in its entirety from the backbone server irrespective of whether its content is eventually viewed by the user.
  - Intuitively, this can be solved by partial access.
    - It is infeasible to develop a generic application-unaware data partitioning technique in the underlying file-system level.
- Goal
  - Design an application unaware content-partitioning strategy
    - The only solution is at the graphical level (i.e. content-partitioning at the output device level).

Design Issues (1)

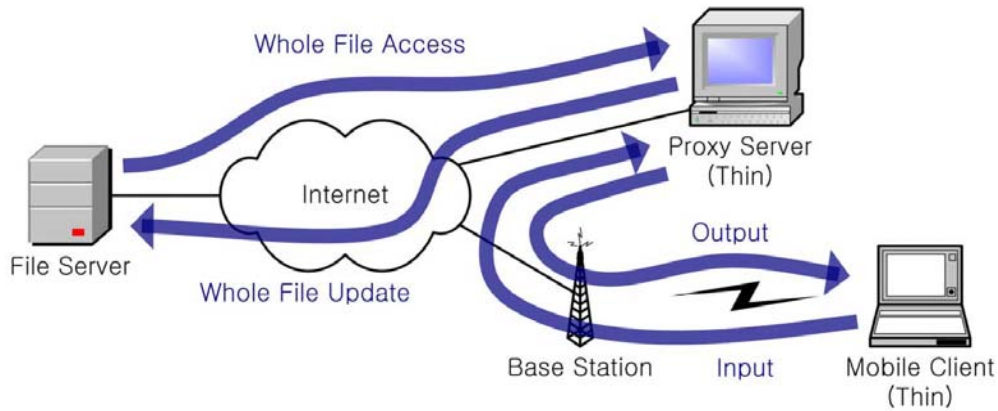
- Usage to fetch-size ratio
  - Graphical content partitioning is highly efficient especially when the byte- or pixel-size of the entire content is large compared to the amount of the file that the user actually views.
  - However, for *highly compressed* content, it may not be efficient in terms of transfer size due to the limitation of real-time recompression.
  - ➡ Hence, it should be used selectively along with traditional models.
- Response time
  - Content partitioning enables quick initial fetch of content due to the smaller byte-size of initial graphical content.
  - However, when the user accesses the entire content of a specific content type, full transfer may be better in average access time.
  - ➡ Thus, it should be supplemented with the binary-content transfer.

Design Issues (2)

- Partial download disconnections
  - Content partitioning reduces the transfer size by enabling partial access and also decreases the probability of network disconnections stalling data transfer.
  - However, even small size partitions may suffer from transmission failures caused by long-scale disconnections.
  - ➡ Therefore, re-usability of partially downloaded graphical content is essential to serve user requests during disconnections.
- Greedy fetch problem
  - Reduction of the greedy fetching size decreases the peak load duration of the system.
  - However, for future disconnected operations, a full binary file may need to be fetched.
  - ➡ To minimize the effect of this non-urgent transmission, it is useful to differentiate the greedy transmission for the initial part from the non-greedy or low-priority transmission for the remaining part.

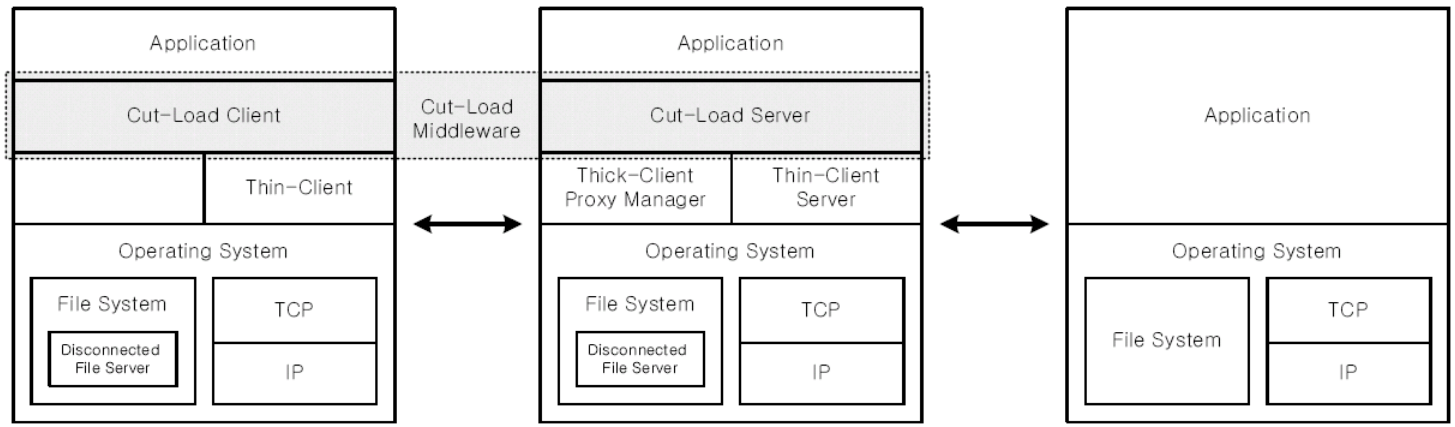
Thin-Client Computing

- Thin-client computing performs *graphical content partitioning*.
  - The proxy server performs all the tasks on behalf of the client.
    - Three-tier architecture: data storage(file server), processing/presentation_lower(proxy), and presentation_higher(client)
    - The only communication between the proxy and the client is for conveying user input and dumping screen output.
    - Formally, thin-client computing involves the use of a simple terminal or processing device connected to powerful proxy servers.
  - Thin-client computing provides the required abstraction for application-unaware and user-activity aware content-partitioning mechanisms in the graphical domain.



Cut-Load Overview

- *Cut-Load*
  - It is an application-unaware access scheme that performs user-activity aware reads for content in a graphical domain.
  - We use the content-partitioning mechanism used by thin-client computing in the Cut-Load middleware.
  - Cut-Load resides at both the client and the proxy as a middleware and hence it is easily deployable.

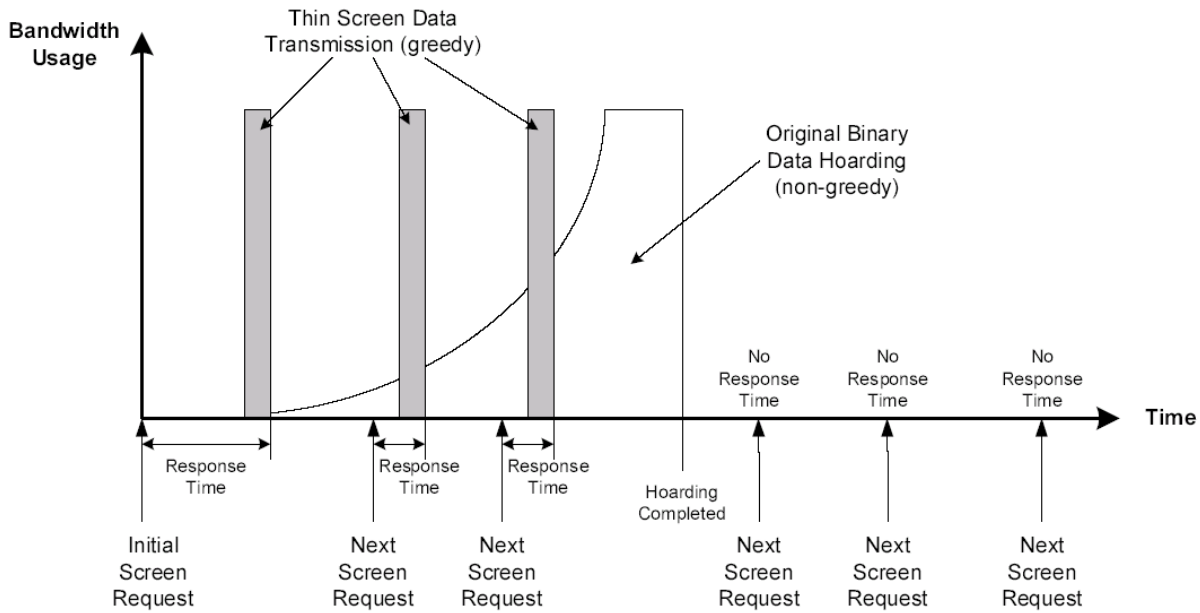


Two Operation Modes in Cut-Load

- *Normal-mode*
  - The client works as a traditional access client.
    - A required content file are fully transferred to the client.
    - The client accesses the fetched file by running a local application.
    - It can perform off-line (disconnected) operations with the cached file.
- *Dual-mode*
  - Initially, the client operates in thin-mode.
    - While a user is seeing the content in the initial thin-mode, the client performs hoarding of original content files in background.
    - When the content file is hoarded completely, the client notifies the user about the change of mode to thick-mode.
  - When hoarding is completed, it performs mode transfer to thick-mode.
    - Then, it opens the hoarded thick-data using an associated application and moves the current system focus to the application window.
    - It closes the remote application that the user used before in thin-mode.

Dual-Mode Operation in Cut-Load

- Performance improvement
  - The response time experienced by the user is decoupled from the actual fetch time for the thick-content,
  - The peak hoarding rate is reduced and hence the system-wide utilization is improved.



Cut-Load Design Elements

- In order to support these mode operations, Cut-Load consists of three basic elements.
  1) Dynamic mode selection
     - It selects the best computing mode for the type of requested content in terms of user performance.
  2) Opportunistic hoarding
     - In dual-mode, it fetches thick-data in a non-greedy fashion in background while fetching thin-data in a greedy fashion.
  3) Transparent mode transfer
     - It allows the mobile client that accesses documents to switch the current access mode if necessary.
- These elements bring benefits of faster access speed and efficient network bandwidth utilization.

Dynamic Mode Selection (1)

- Thin-friendliness
  - Real-time compression processing in thin-client computing may result in *significant bandwidth inefficiency* for some types of pre-compressed content.
    - We call them thin-unfriendly content types.
    - MP3, MPEG, animated picture, etc.
  - Reason of thin-unfriendliness
    - Poor re-compression performance in thin-mode
    - Wrong selection of a re-compression method
- Content pixel-size
  - Content access with small fetch-size ratio shows better bandwidth-efficiency in dual-mode.
    - Pixel-size information of content can be provided by a content container interface function that every application instance has.
      - The interface gets the pixel-size by measuring the pixel-size of scrollable area in the content container.

Dynamic Mode Selection (2)

- Network connectivity
  - Interactive operations in thin-client computing are based on strong connection between a client and a server.
  - The client estimates the current long-term signal-to-noise ratio (SNR) first to check eligibility for thin-client solution.
  - If the current connection is not strong enough for dual-mode, the client accesses the content in normal-mode regardless of other decision factors.
- Decision Heuristics
  - Decision factors
    - *Network connectivity, byte-size, pixel-size, thin-friendliness*
  - Optimal threshold values of these factors are dependent on what and how a user accesses.
    - Data mining of user access patterns is necessary for better performance.

Opportunistic Hoarding

- Low priority transmission
  - It minimizes the impact on the thin transfers that use normal TCP.
  - It uses weighted additive increase multiplicative decrease (W-AIMD) congestion control
    - It controls the weight to set a specific fraction of the bandwidth obtained by the high-priority thin-client flows that use normal TCP.
    - When the hoarding starts, the flow is assigned aninitial weight of 1/w.
- Adaptive transmission
  - As the user accesses more screens staying in the same content, the probability that the user will see the whole content increases
    - It increases the hoarding rate to reduce the future response time.
- Advantages:
  - It avoids the starvation of hoarding flows so that after a sufficient time even in the presence of normal TCP flows, the raw content would be hoarded and be ready.
  - The exponential rate adaptation would ensure that sufficiently long hoarding flows achieve data rates comparable to normal TCP flows and complete the hoarding faster.

Mode Transfer

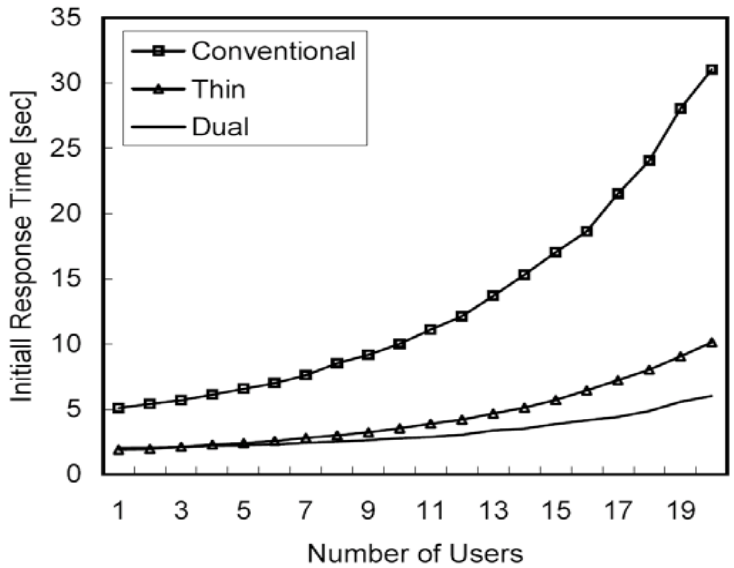
- Basic operations
  - If the client is still accessing the same content in dual-mode when opportunistic hoarding is complete.
  - It stops the current thin operation and notifies the mode transfer to the user.
  - When the transfer is completed, it begins to provide access to the hoarded content file in normal thick mode.
- Cut-Load provide a seamless user access after mode transfer.
  - Environmental synchronization
    - System settings: *screen resolution, keyboard layout, clipboard content*
    - Application settings: *menu bar, zoom rate, view option*, etc.
  - Focus synchronization
    - Mouse focus: current location of the mouse cursor
    - Keyboard focus: current position of the keyboard input
    - Screen focus: the screen position of the client area in the document layout
    - Once the new local application is opened, the captured focuses are restored by OS-specific interface functions.

Simulation Setup

- System Parameters
  - Environmental setup
    - Screen resolution: 1024x768, Client area size : 1006x511
    - Cell capacity: 640 Kbps
  - Document characteristics
    - Byte-size distribution of document files in thick-mode
      - Single-side tail-less Gaussian distribution (mean:400 KB, STD:200 KB)
    - Byte-sizes distribution of each screen in thin-mode
      - Single-side tail-less Gaussian distribution (mean:150 KB, STD:50 KB)
    - Average number of screens per document: 1.468 screens
  - Operation setup
    - Byte-size threshold for mode selection : 300 KB
    - Probability of accessing the next screen in the same document
      - Initial screen : 40%
      - After each next screen access: increased by 10% (up to 90%)
    - User access interval
      - Single-side tail-less Gaussian distribution (mean: 20 s, STD: 10 s)

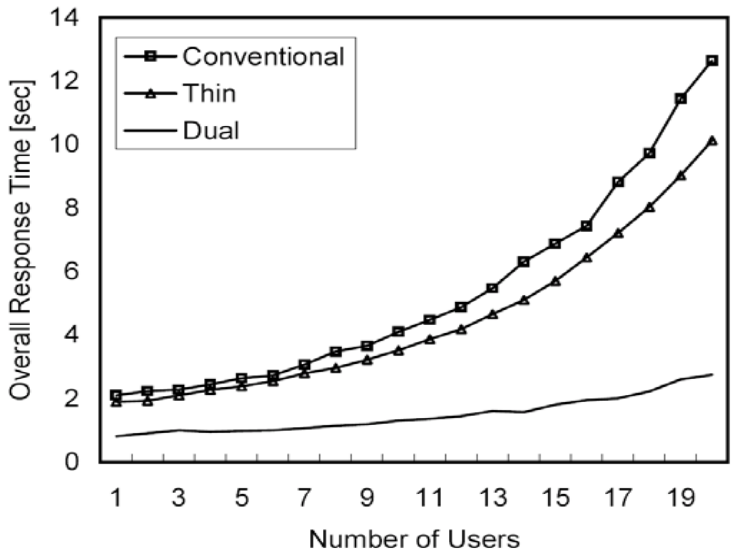
Response Time Performance (1)

- Initial Response Time Performance
  - Initial response time in normal-mode is significantly large.
    - However, once the whole file is fetched, additional accesses are provided instantaneously.
    - It show an exponentially increasing pattern when the number of user increases.
  - The dual-mode client shows better performance than others.
    - The performance of both the thin- and dual-mode client is not degraded significantly.
    - By the benefits of hoarding, other dual-mode clients don't spend bandwidth continuously as other thin-clients, and thus the total greedy transmission size is relatively smaller.



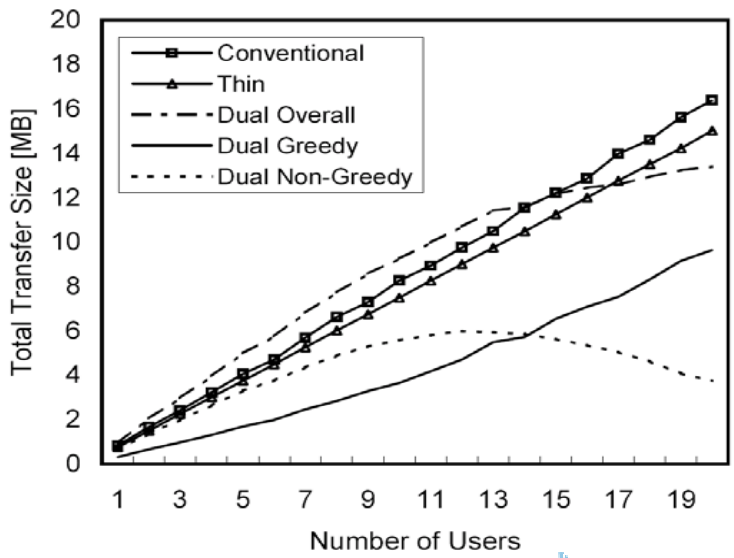
Response Time Performance (2)

- Overall Response Time Performance
  - The thick-(conventional) and thin-mode client show an exponentially increasing pattern when more users share the network bandwidth.
  - The dual-mode performance is not affected significantly by the number of users.
    - The dual-mode client shows relatively stable response time under 2 seconds.
    - The dual-mode consumes much less bandwidth in greedy transmission than other modes.
    - The non-greedy hoarding doesn't have a negative influence on other users' response time performance.



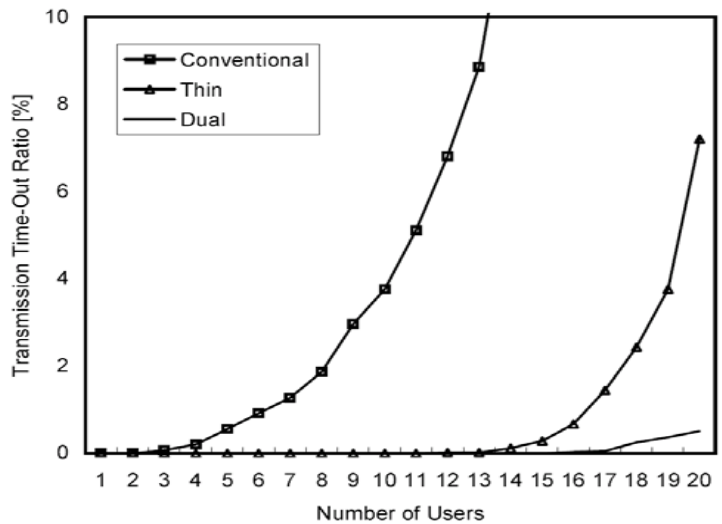
Transfer Size Performance

- The overall transfer size in dual-mode client is larger or similar, however its greedy transfer size is about a half of others.
  - The dual-mode operations decouples the greedy transmission for the urgent part required immediately from the non-greedy transmission for the less urgent part of content.
- The non-greedy transmission has a peak point around 23.
  - The available excess bandwidth for the dual-mode operations begins to decrease after the overall network utilization becomes saturated.
  - When the size of non-greedy transmission becomes almost zero due to the extremely heavy traffic, the transfer size overhead of the dual-mode becomes the same as that of the thin-mode.



Hoarding Performance

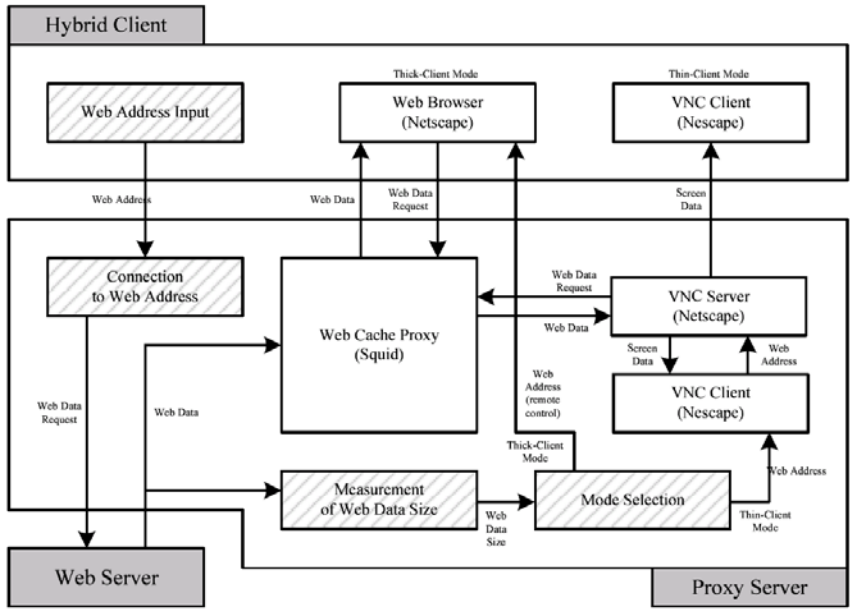
- We modeled system utilization as the transmission time-out rate ratio.
  - It is the ratio of the total number of connections which have response times greater than the average user tolerance level.
- Opportunistic hoarding used in dual-mode performs better than pure greedy transfers used by traditional systems.
  - The lower peak rates of opportunistic hoarding improves the system utilization.
  - Even though the increased background traffic also affects the main greedy transmissions more, the dual-mode can keep dominance in the response time performance.



Prototype Straw-man

- Prototype
  - Linux platform with the SQUID web proxy cache
  - VNC as the thin-client computing solution.
  - Two virtual panels representing the thick- and thin-client modes.
  - Netscape web-browser
    - Remote control command-line function

User Access	Cut-Load Decision	Behavior Results
HTML / 3 PageDowns	Thick-Mode	Success
DOC / 6 MouseClicks	Dual-Mode	Success
PDF / 12 PageDowns	Thick-Mode	Success
HTML / None	Thick-Mode	Success
HTML / 2PageDowns	Thick-Mode	Success
PDF / 50 PageDowns	Dual-Mode	Success
DOC / 4 PageDowns	Dual-Mode	Failure
HTML / 1 MouseClick	Thick-Mode	Success
Total		87% Success Rate



Summary

- Conventional client-server models are not optimal for low-bandwidth wireless networks because of their user-activity unaware operation in the file level.
- The proposed strategy uses a intelligent mix of binary file-transfers and graphical content-partitioning along with features such as opportunistic hoarding to reduce the bandwidth consumption as well as response times for web-access.
- We evaluated the performance of the Cut-Load strategy and proved its benefits over traditional access systems.
- We also analyzed the behavior of the real-life implementation of the Cut-Load prototype.