

# Client-Side Web Acceleration for Low-Bandwidth Hosts

---

Tae-Young Chang<sup>1</sup>, Zhenyun Zhuang<sup>1</sup>  
Aravind Velayutham<sup>2</sup>, and Raghupathy Sivakumar<sup>1</sup>

<sup>1</sup>Georgia Institute of Technology, Atlanta, GA, USA

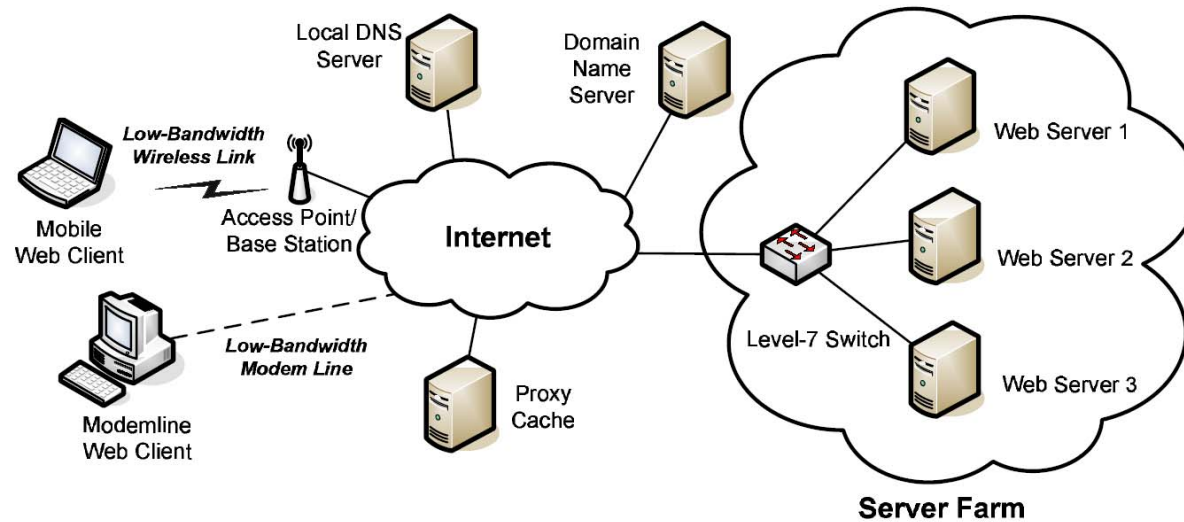
<sup>2</sup>Asankya Inc., Atlanta, GA, USA

# Introduction

---

- Current Web optimization techniques
  - Web cache proxies, persistent connections (HTTP/1.1), content distribution networks (CDNs), WAP, BREW, etc.
  - Web browsers **still suffer** in low-bandwidth conditions.
    - Current fetching model of Web browsers is **not optimal** in low-bandwidth networks.
- Our contributions in this paper are:
  - Identification of the **problems that lead to inefficiency** of current Web browsers by carefully analyzing the interactions of several factors.
    - Screen contention problem
    - Bandwidth under-utilization problem
  - Proposal of **three mechanisms** to reduce Web response time in an *easy-to-deploy* fashion.
    - Prioritized fetching (PF)
    - Object reordering (OR)
    - Connection management (CM)

# Current Web Access Model (1)



- 1) **Load balancing** can be performed by multiple Web servers.
  - A *layer-7 switch* dynamically **rewrites** domain names of objects in HTML.
  - Objects requests for a single Web page are delivered to multiple servers.
- 2) **Multiple connections** can be opened to a single Web server by browsers.
  - Internet Explorer and Netscape Navigator open up to 2 and 6 TCP connections to a single server, respectively.
  - A *parsing engine* in each browser inserts object requests to the message queues of the connections in a **round-robin** fashion.

## Current Web Access Model (2)

		HTML	IMG	Others	Total
Byte-size per object [KB]	Mean	31.72	2.46	12.91	225.96
	STD	35.51	5.90	9.67	186.04
Number in first screen	Mean	1	17.31	4.41	22.72
	STD		15.36	6.22	16.37
Number in all screens	Mean	1	46.80	3.99	51.79
	STD		28.16	6.22	30.34
Number of web-servers	Mean	1	5.16	1.74	5.50
	STD		2.90	1.20	3.38
Width [pixels]	Mean				998
	STD				46.49
Height [pixels]	Mean				1937
	STD				1119

- Measurement of *comScore's Top 50 Web Sites*
  - Default full screen resolution: 1024 x 768 [pixels]
  - Pixel size of the client area in Internet Explorer: **1006 x 511** [pixels]
    - *Client area*: an effective area for displaying a Web page in a browser
    - *Screen*: a unit of an area, 1006 x 511 [pixels]
    - The size of a screen is equal to that of the client area.
  - Average number of screens per Web page:  $1937\text{pixels} / 511\text{pixels} = 3.7$

# Screen Contention Problem (1)

---

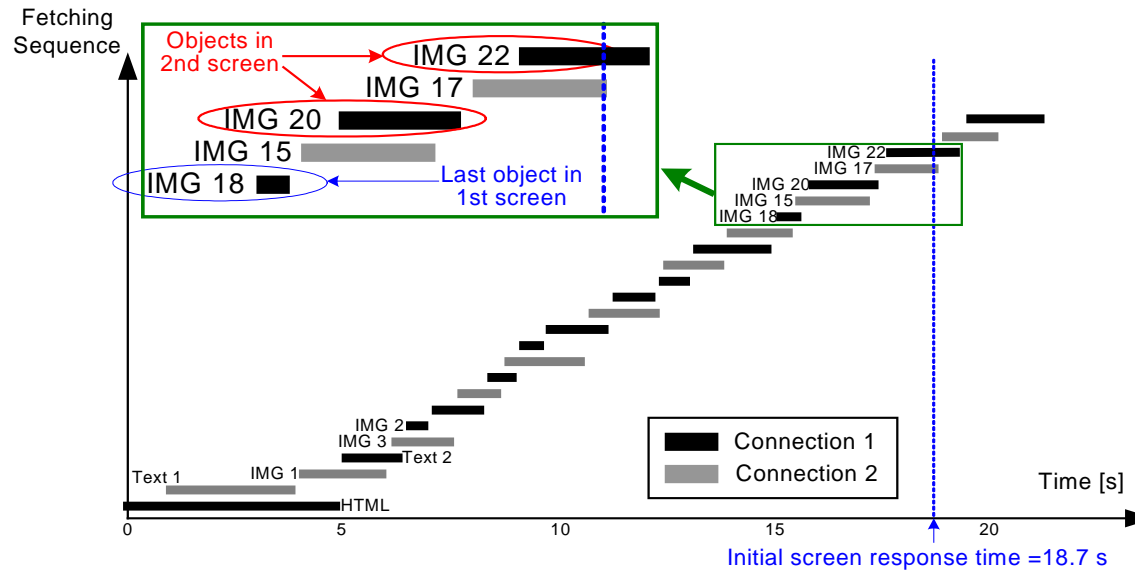
- What is **Screen contention**?
  - 1) Current Web browsers always perform **greedy** fetching.
    - They fetches the **entire** objects of a Web page, regardless of necessity.
  - 2) Users always perform partial Web-page accesses at any instant.
    - Only a **part** of a Web page can be accessed at a time, through the client area.
    - Users may not scroll down through the entire content of a Web page and leave.
  - Thus, fetching off-screen objects is not always necessary.
    - Fetching of necessary on-screen objects may be slowed down.
    - Objects from different screens compete for bandwidth: **screen contention**
- Why does screen contention occur?
  - Web browsers adopt an **imperfect fairness** model.
    - Browsers considers only fairness in the **number of objects per connections**.
    - Object requests are inserted to multiple connections in a **round-robin** fashion.
  - **Disparity of cumulative transfer size** among multiple connections
    - Connections having only small-sized objects may finish on-screen transmission early and begin to fetch off-screen objects.

# Screen Contention Problem (2)

---

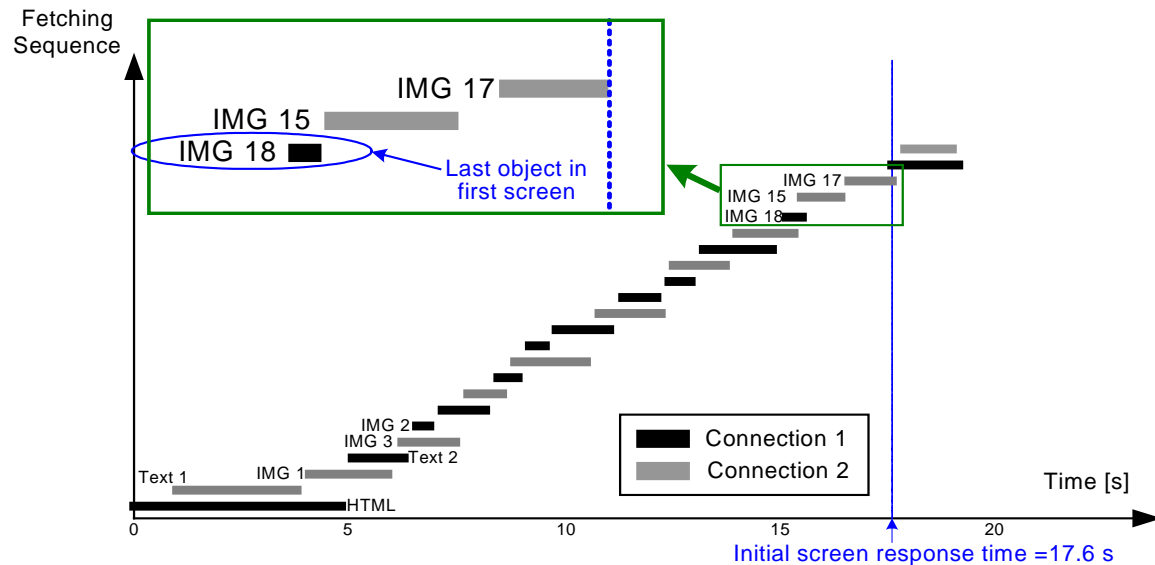
- Simulation setup
  - ns2 network simulator
    - Reno-FullTCP package: support for bi-directional transmissions
  - Link characteristics
    - Link between Web client and backbone network
      - 100-kbps bandwidth and 100-ms link delay.
    - From DNS/Web servers to backbone network
      - 1-Mbps bandwidth and 5-ms link delay
  - Browser Characteristics
    - The same Web statistics of *Top 50 Web Sites* are used.
    - Processing time per object: 200 ms.
    - HTTP request message size: 500 B.
    - HTTP/1.1 persistent connection: supported, pipelining: *not* considered.
    - Cache function: disabled.
  - Metric
    - Initial screen response time
      - Time spent until all objects for displaying the initial screen are downloaded completely

# Screen Contention Problem (3)



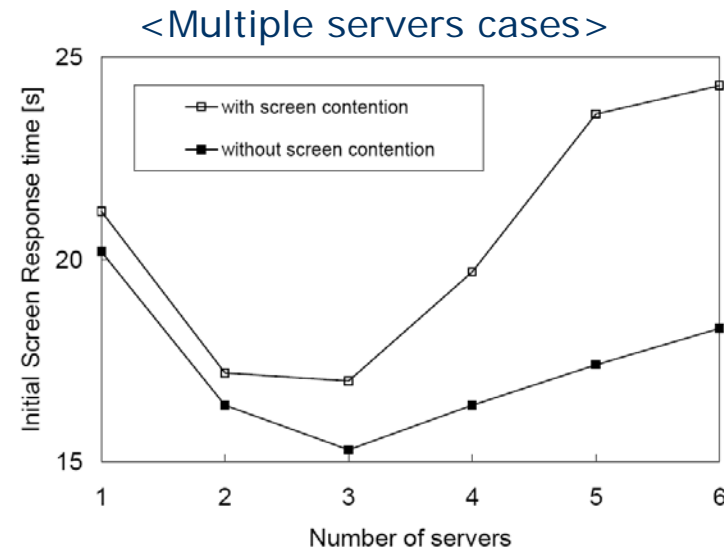
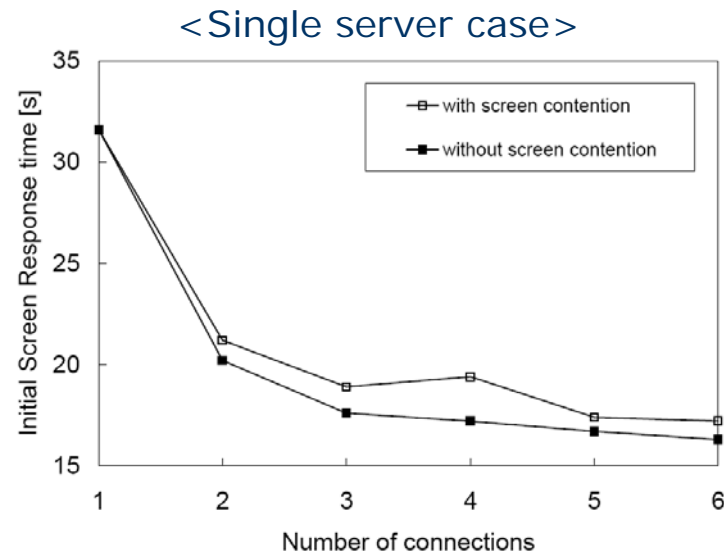
- In this simulation,
  - All the objects are from a single server.
  - The initial screen has 18 on-screen objects.
- Screen contention scenario
  - Two off-screen objects are fetched before the initial screen is fully displayed.
    - Fetching **unnecessary** objects consumes some portion of bandwidth
    - The resulting response time for initial screen is increased unnecessarily.

# Screen Contention Problem (4)



- Ideal scenario
  - An intuitive solution is to prevent unnecessary object fetching.
    - When a faster connection completes all on-screen object transmissions, it stops fetching and waits for the other connections to finish fetching.
  - The remaining connections can obtain **more** bandwidth.
    - The response time for the current screen can be minimized.

# Screen Contention Problem (5)



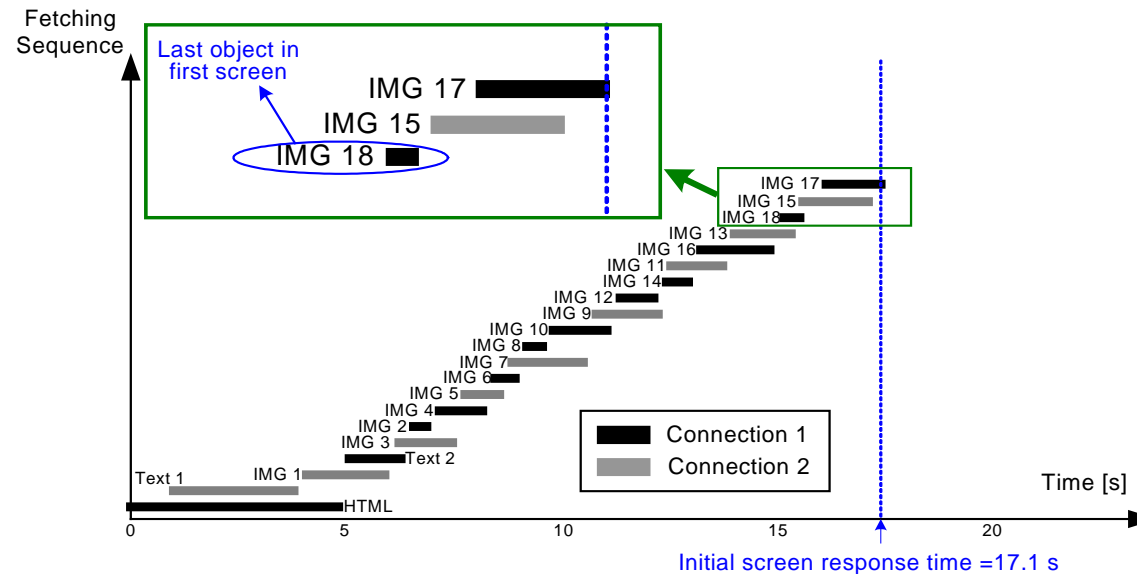
- Single server case (no load balancing)
  - The performance of both is not directly affected by the number of connections.
  - Number of connections = 3: the performance improvement is saturated.
- Multiple servers case (load balancing, up to 2 connections to server)
  - Number of servers > 3: the performance of the conv. Browser is degraded.
    - The degree of contentions among connections to different server becomes higher.
  - The performance in the ideal browser is **less** influenced by the number of servers.

# Bandwidth Under-utilization (1)

---

- What is **bandwidth under-utilization**?
  - Idle time of a network increases with the decrease in the number of simultaneous active TCP connections.
    - An optimal number of simultaneous TCP connections exists.
  - A non-optimal number of active connections result in under-utilization of links.
    - We refer this to as the **bandwidth under-utilization** problem.
- Why does bandwidth under-utilization occur?
  - Current browsers do not maintain the optimal number active connections.
    - Only a small number of connections are active at any instant in a browser.
    - Other connections go into the idle status after completing all their object fetching.
  - In Web browsers, bandwidth efficiency is determined by how much the ending times of transmissions in all connections are **synchronized**.

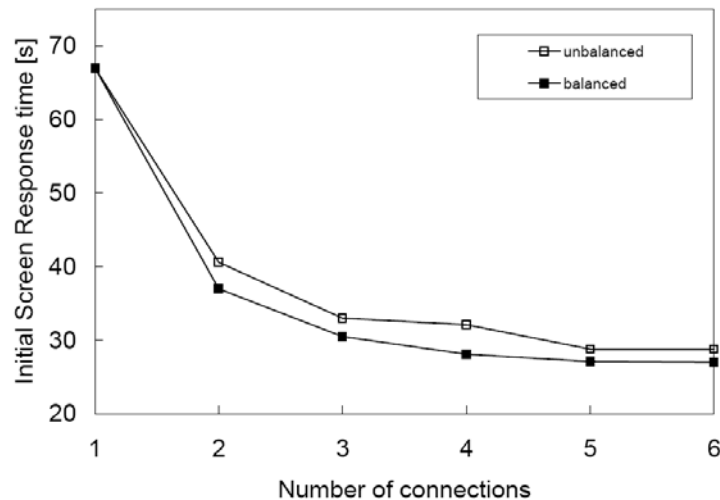
# Bandwidth Under-utilization (2)



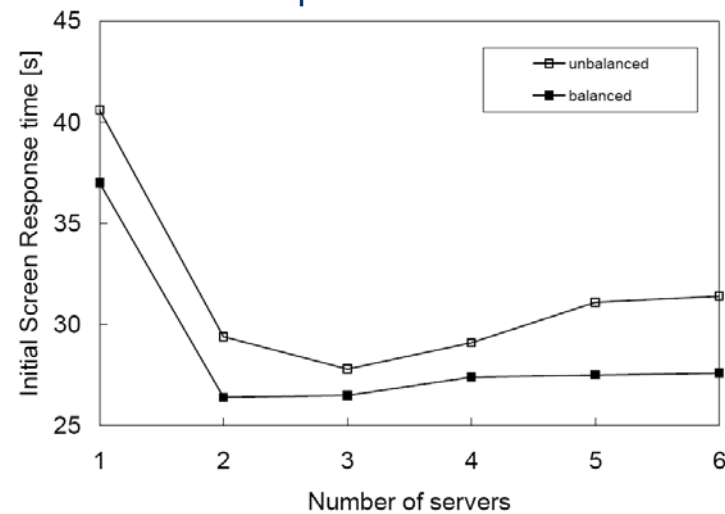
- Ideal case
  - The intuitive solution is to schedule different object requests across multiple servers such that as **many** connections are active as possible.
    - Each TCP connection should have at least one pending request in the message queue.
    - An inactive connection **takes over** the unfulfilled object requests from others.
  - Cooperative connections can use bandwidth more efficiently and improve the initial screen response time.

# Bandwidth Under-utilization (3)

<Single server case>



<Multiple servers case>



- In the simulations,
  - screen contention does not exist in this scenario.
- Single server case
  - The performance of both is not directly affected by the number of connections.
- Multiple servers case
  - With 2 or 3 (probably the optimal numbers) servers, both show the best performance.
    - As the number of servers increases beyond 3, performance rather becomes degraded.
  - The ideal scheme shows up to a 20% performance improvement.



# Prioritized Fetching (PF) (1)

---

- Basic operation steps
  - 1) Obtain the initial screen view information in the entire document layout
  - 2) Prioritize embedded objects according to their locations in the layout
  - 3) Perform fetching objects according to their priority levels
    - When a user scrolls to a different view, repeat the process again.
- Initial object prioritization
  - Object-type-based prioritization
    - Text-based objects: HTML, javascript, cascading style sheets, etc.
      - They play an important role to construct the **overall HTML display layout**.
      - PF Gives the **highest** priority to these objects.
    - Other object types: Image (IMG) and multimedia objects, etc.
      - Different priority levels are given to objects according to their **locations**.
      - The **highest** priority level is assigned to **on-screen** objects.
  - Location-based prioritization
    - Detection of pixel size of objects
      - An HTML document file generally defines the pixel size of image objects.
      - In cases of no pixel size, PF uses an **averaged value** based on browsing history.
      - A Web browser can construct the full page layout without downloading these objects.

# Prioritized Fetching (PF) (2)

---

- Detection of location of objects
  - PF scans the document object model (DOM) tree. When it finds a target object, it
    - 1) searches all the successors in the tree
    - 2) calculates location offsets from successors to predecessors
    - 3) repeats the process until it reaches the top of the tree.
  - The absolute location in the layout is the **sum** of all the relative offsets.
- **Selective object fetching**
  - In HTTP, priority-based bandwidth allocation is not possible.
    - Connections perform **short bursty on-off transmissions** by sharing bandwidth.
    - A single TCP connection cannot send **both** high- and low-priority objects.
  - PF uses a delayed-transmission scheme.
    - PF inserts request messages into the already-in-use queues.
    - Low-priority objects begin to be fetched only after all the higher-priority queues become empty.
- **Re-prioritization**
  - When the screen focus is **moved to a new area**, PF re-prioritizes all the remaining objects in the queues for the newly focused area.
    - PF keeps the currently incoming transfers.

# Objects Reordering (OR) (1)

- Basic operation steps
  - 1) Execute an initial assignment of objects
  - 2) Perform a TCP-aware ordering of objects
  - 3) Perform dynamic objects rescheduling
- Initial objects assignment
  - Conventional browsers perform byte-size-unaware round-robin assignments.
    - It causes unsynchronized ending times among different connections, and thus increases response time.
  - Initially, OR performs load balancing among connections by distributing the same amount of objects to every connection.
    - Expected ending time is given by  $SizeData/BW_{available} + n*rtt/2 + T_{Proc}$ 
      - $n$  is the number of objects, and  $T_{Proc}$  is the processing time.
      - The first term **dominates** over other terms in low-bandwidth networks.
    - OR estimates the ending times by considering both the objects' pixel-size included in HTML document and the object formats, such as gif and jpg.

# Objects Reordering (OR) (2)

- TCP-aware objects reordering

1) In TCP, appropriate ordering can minimize the adverse effect of **slow start**

- 7-3-2 KB objects fetching takes 5 *rtts*. (2+4+1+3+2 KB) (if *cwnd* starts from 2)
- 2-3-7 KB objects fetching takes 3 *rtts*. (2+3+7 KB)

2) Small objects can be rescheduled in a **finer** granularity.

- With small objects being put at the end of connections, it is more likely to reschedule objects among connections.
- Thus, OR orders the fetching sequence in a *rats-elephants-rats* fashion.
  - Data-size-based sorting
  - Size-based round-robin assignments

- Dynamic objects rescheduling

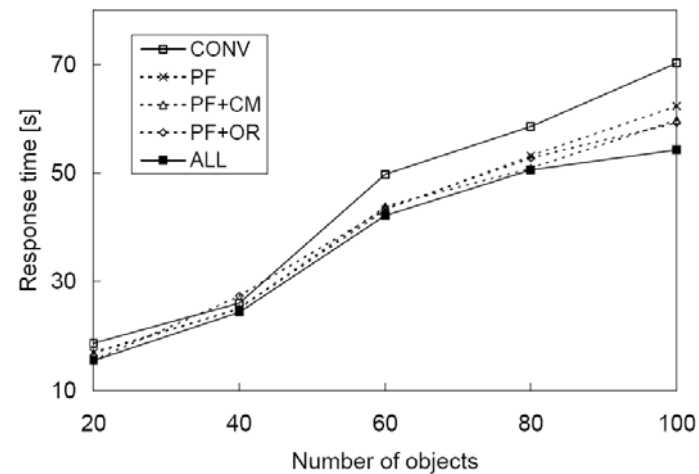
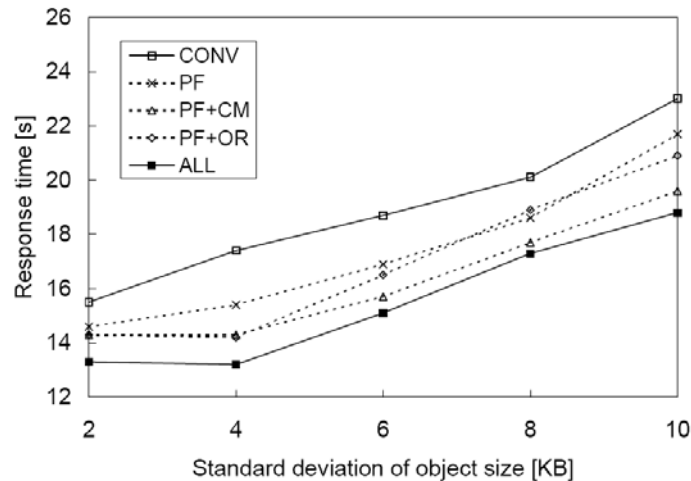
- Because of dynamic behavior of TCP connections, the total fetching time of different connections may still vary significantly.
- OR dynamically reschedules objects from the busy connections to the idle ones in an on-demand fashion.

# Connection Management (CM)

---

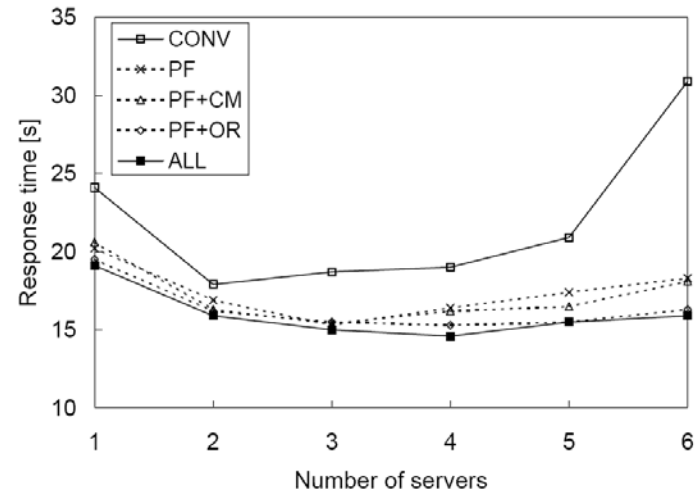
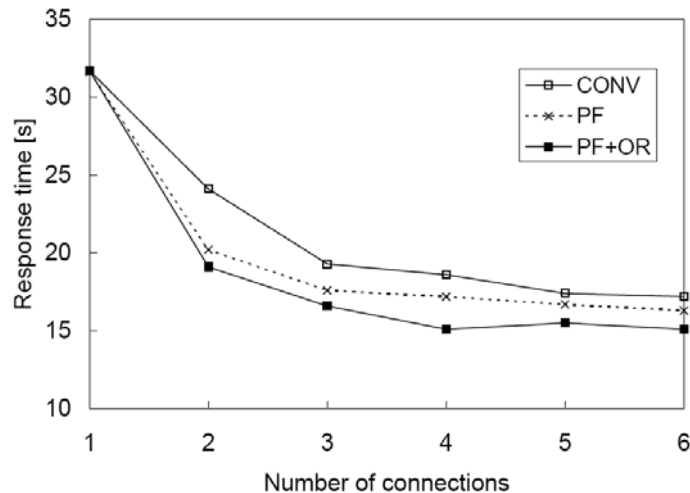
- Basic operations step
  - 1) Estimate per-connection load
  - 2) Adjusting the number of connections for each server
- Per-connection load estimation
  - Estimation of the ending time of downloading
    - CM uses the byte-size information that OR converted earlier.
- Dynamic connection assignment
  - CM assigns more connections to servers with larger data size and less connections to servers with smaller data size.
    - The total number of connections is always maintained the same as in current browsers for friendliness and compatibility.
      - When it assigns one more connection to a server, one less connection should be deducted from some other server.
    - CM limits the maximum number of connections to a server to 4.
      - Allocating too many connections to the same server does not necessarily lead to better performance.

# Performance Evaluation (1)



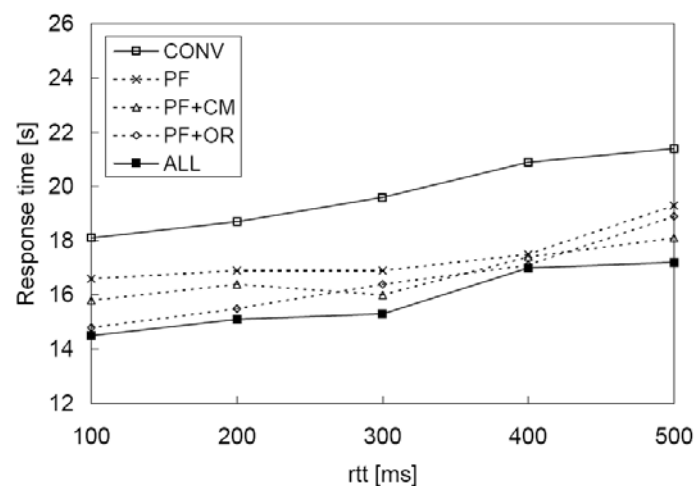
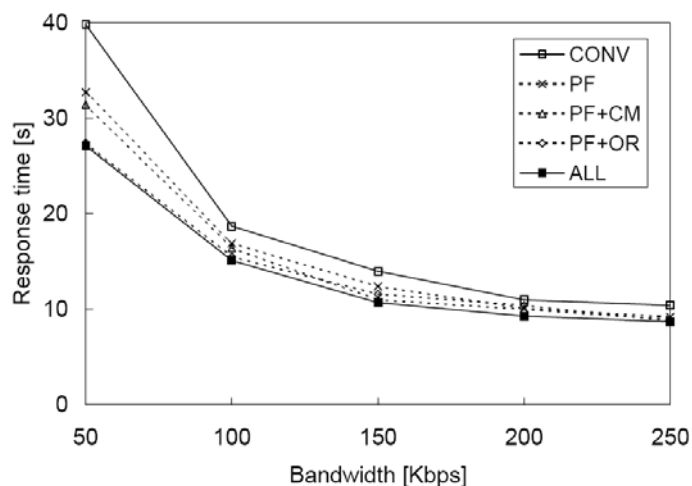
- Impact of object characteristics
  - Variance of object size
    - **Large** variance of sizes reduces bandwidth utilization.
  - Number of objects in a Web page
    - As more objects are included in a page, larger response time is expected.
  - Our solution reduces up to 30% of response time.

## Performance Evaluation (2)



- Impact of number of connections/servers
  - Number of connections
    - Number of connections > 4: no obvious performance improvements in both
  - Number of servers
    - Number of servers > 3: performance begins to be degraded.
      - Increasing the number of servers does not necessarily always result in better performance
  - Up to 20% of response time can be reduced by using our solution.

# Performance Evaluation (3)



- Impact of network characteristics

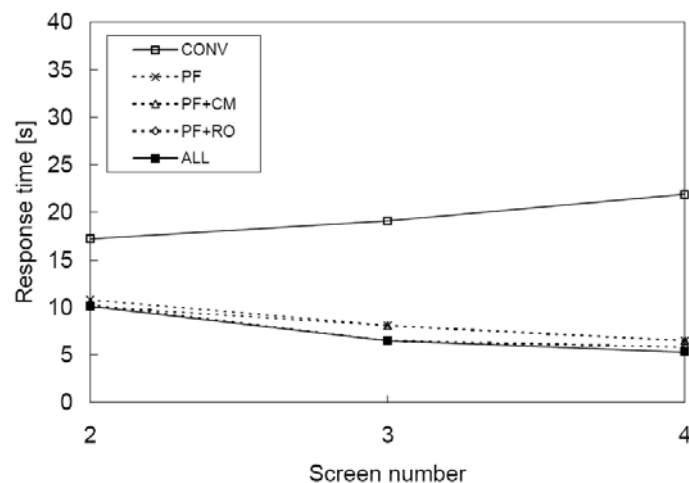
- Bandwidth

- Our solution brings more performance improvement for smaller bandwidth.
      - Smaller bandwidth makes the screen contention problem more severe.

- Round-trip time (*rtt*)

- Our solution reduces the required number of *rtts* for object transmission.
  - Around 20% performance improvement is achieved by our solution.

# Performance Evaluation (4)



- Impact of fast scroll
  - In conventional browsers, when a user scrolls away from the initial screen, response time for that screen increases significantly.
    - Displaying of any screen requires downloading of **all previous screen(s)**.
  - Our solution has smaller response time, as a user scrolls farther away.
    - PF always fetches the **current screen first** in an on-demand way.
    - In most Web pages, **less** data is located in farther screens.
  - Up to 70% of the response time can be reduced.

# Conclusions

---

- In this paper, we explore the reasons that conventional web access models are not appropriate for low-bandwidth hosts. We identify the **screen contention** and **bandwidth under-utilization** problems, which result in large user-perceived response time.
- To address this problem, we propose a new Web access scheme for low-bandwidth hosts, which uses an intelligent mix of **prioritized fetching**, **object reordering**, and **connection management**.
- Using simulations with the statistics of *Top 50 Web Sites*, we evaluate the performance of our scheme and prove its benefits over conventional Web access models.