CS 525M – Mobile and Ubiquitous Computing Seminar

Josh Schullman &
Ted Goodwin

Broadcast Disk Implementation

- Contents
  - Introduction
  - Design
  - Results
  - Conclusion
  - Future Work

Broadcast Disk Implementation

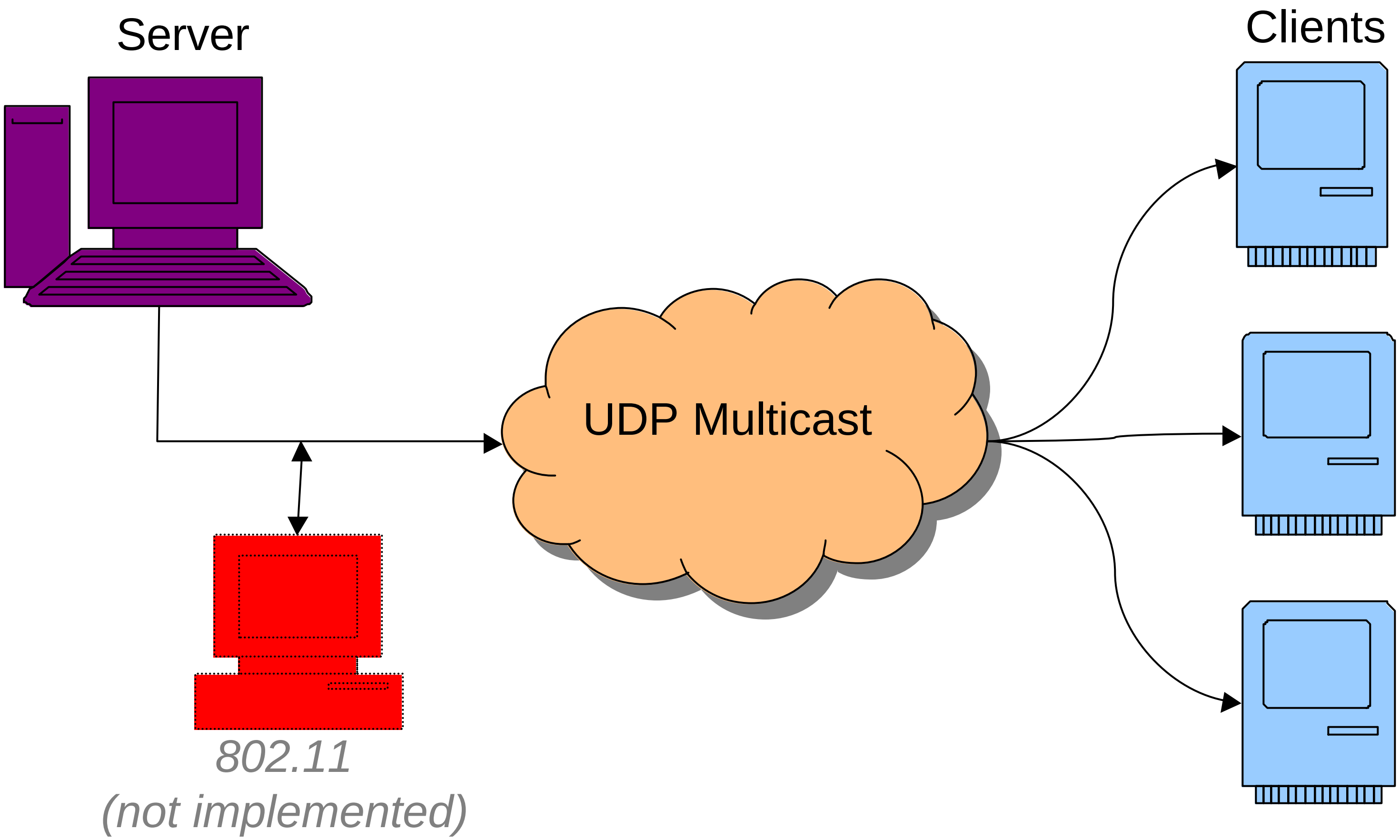
- Introduction
  - Implemented Multicast Broadcast disk
    - Client / Server
    - Cache Policy
    - Broadcast Disk Algorithm
    - Test Script

Broadcast Disk Implementation

– Design
  - Implementation tools
    – UDP Packets
    – Java Networking
    – Eclipse IDE for development

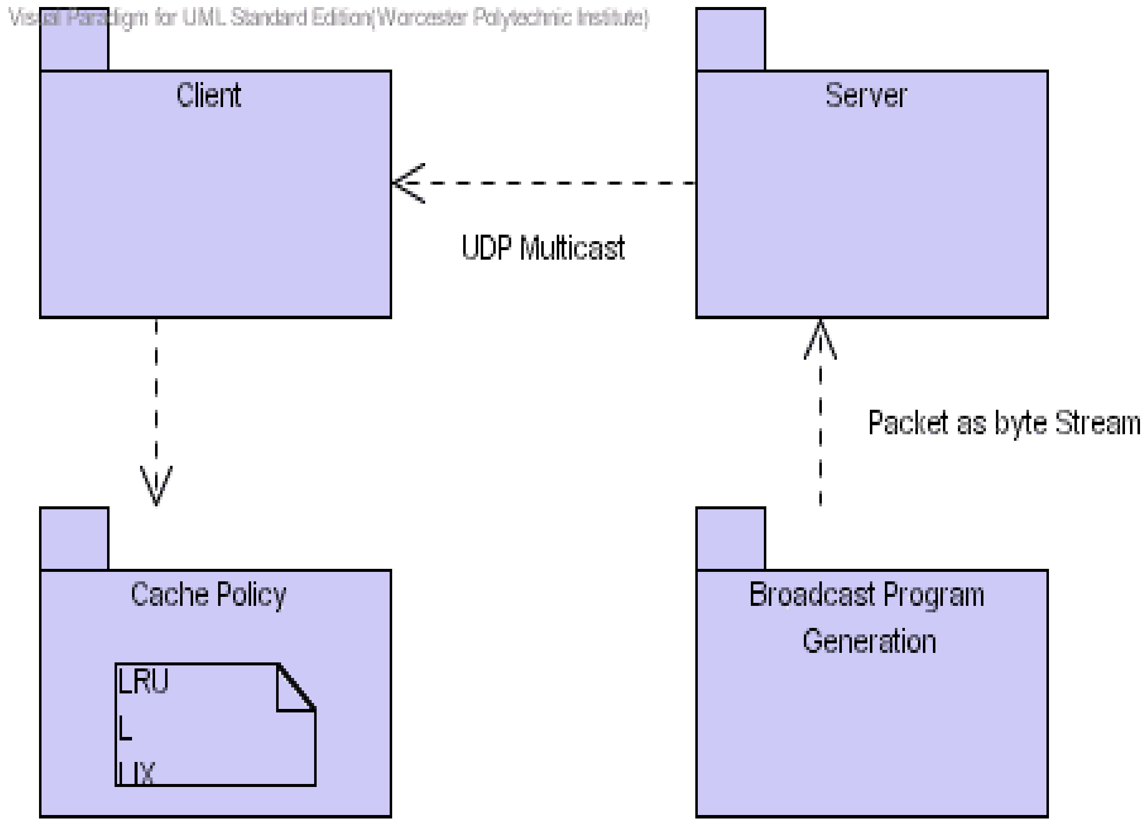
Broadcast Disk Implementation

Network Layout



Broadcast Disk Implementation

Design Diagram



Broadcast Disk Implementation

Design- Cache Policy

- Implemented
  - LRU - linked list accessed pages are moved to the top of the list
  - LIX - Link list for each broad cast disk smallest *lix* value of bottom pages ejected.
  - L - like LIX except same frequency value for each disk
- Not Implemented
  - P – highest access probability in cache
  - PIX – lowest ratio of access probability to broad cast frequency

Broadcast Disk Implementation

Design- Cache Policy

– Lix Example

- $p_i = ? / (currentTime – t_i) + (1 - ?)p_i$
- $lix = p_i$ / frequency of the page

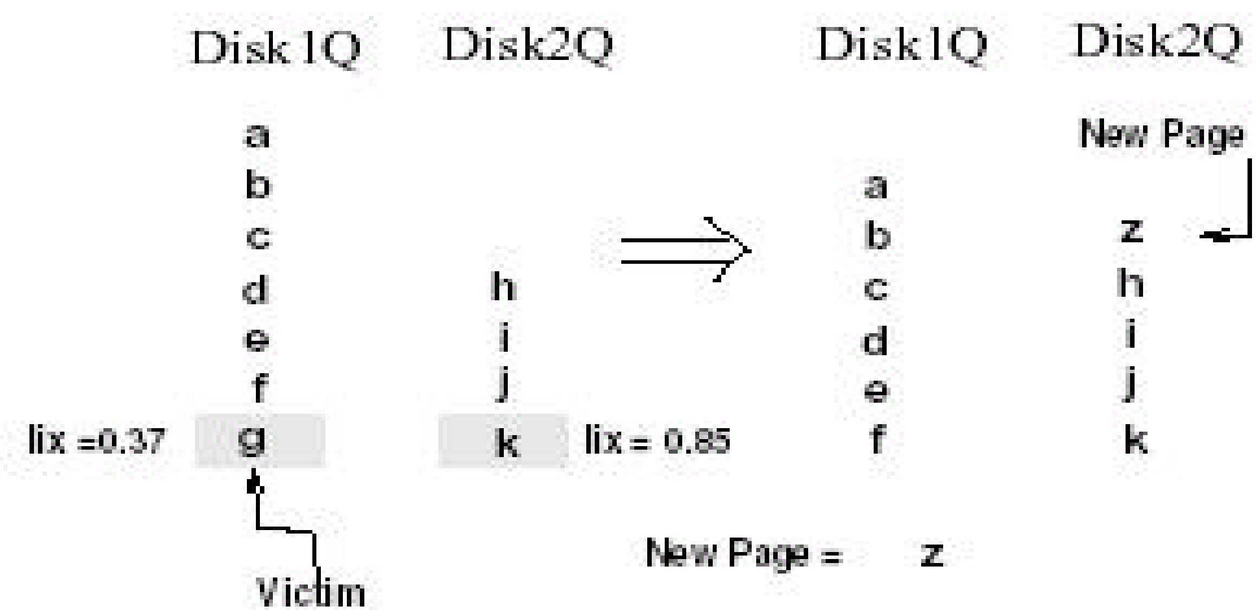


Figure 12: Page replacement in $\mathcal{LIX}$

Broadcast Disk Implementation

Design – Broadcast Program Generation

- Create page vector
- Apply noise values
- Calculates Chunks per disk
- Generate relative frequency per disk
- Interleave chunks while applying offset
- Generate page list

Broadcast Disk Implementation

Design – Client

- Executes as a java thread
- Listens on UDP multicast
- Uses three different caches or no cache
- Receives the following parameters:

CacheSize	Client cache size (in pages)
ThinkTime	Time between client page accesses (in broadcast units)
AccessRange	# of pages in range accessed by client
θ	Zipf distribution parameter
RegionSize	# of pages per region for Zipf distribution

Broadcast Disk Implementation

Design – Server

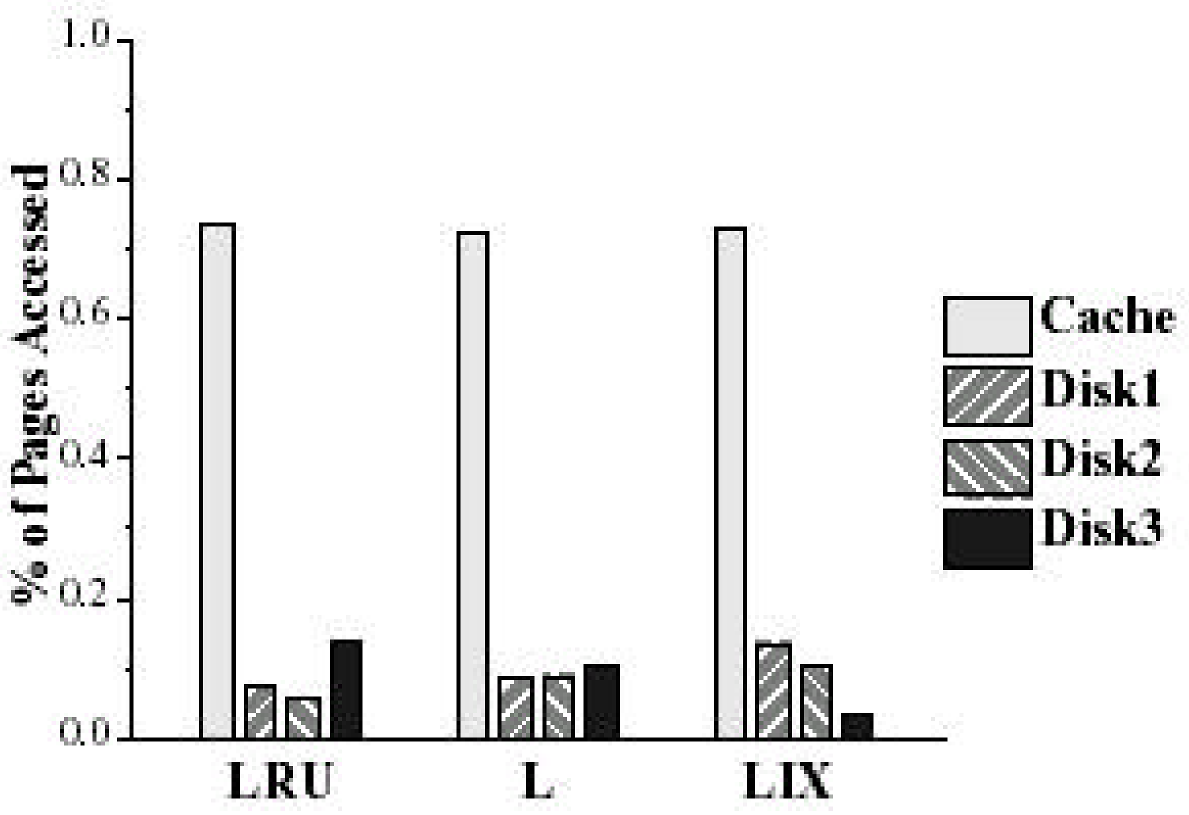
– Executed as a thread.

– Sends UDP packets over multicast

– Receives the following parameters:

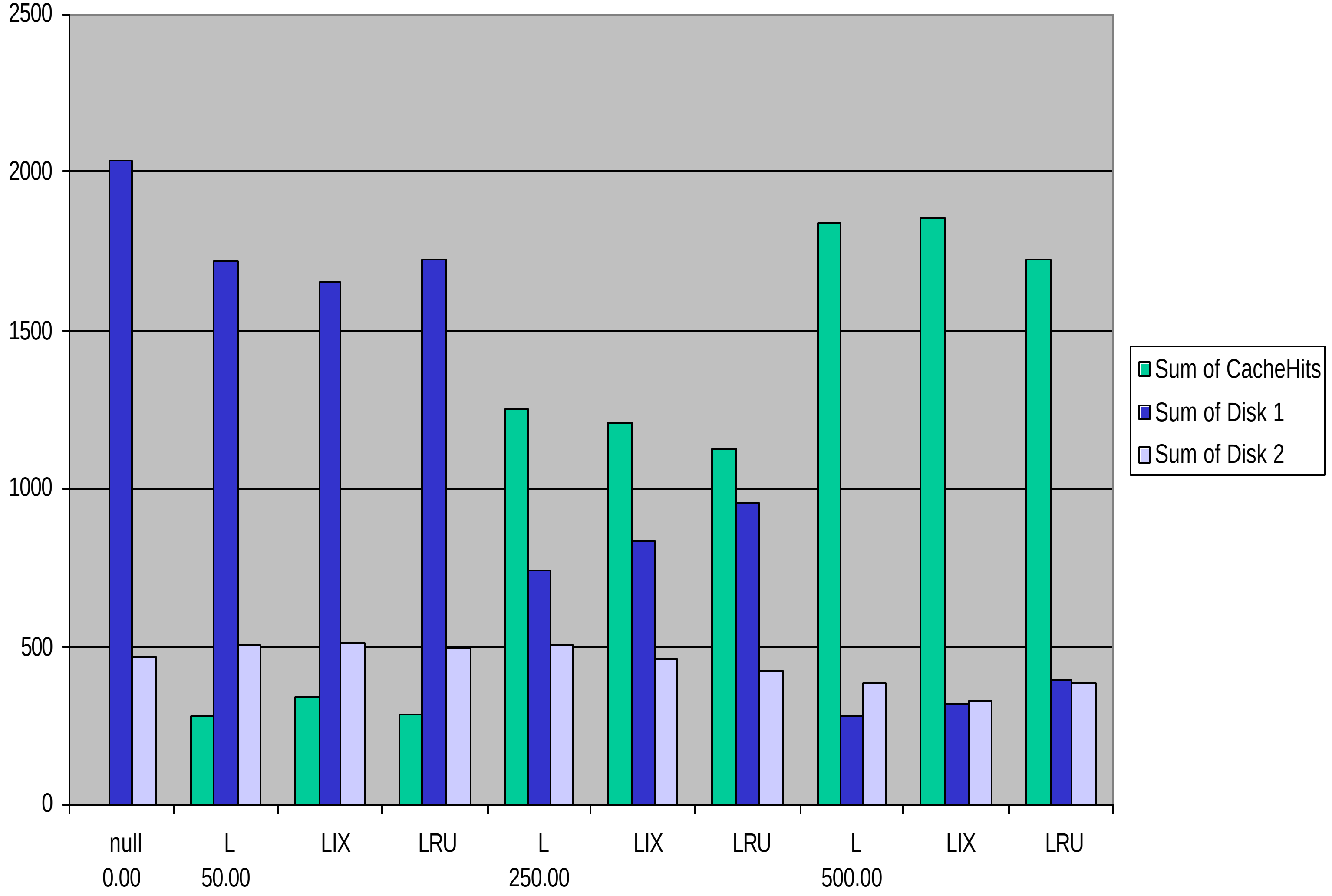
ServerDBSize	Number of distinct pages to be broadcast
NumDisks	Number of disks
$DiskSize_i$	Size of disk i (in pages)
Δ	Broadcast shape parameter
Offset	Offset from default client access
Noise	% workload deviation

Broadcast Disk Implementation



Broadcast Disk Implementation

<500,4500>, 0 offset, 0% noise, delta(5)



Broadcast Disk Implementation

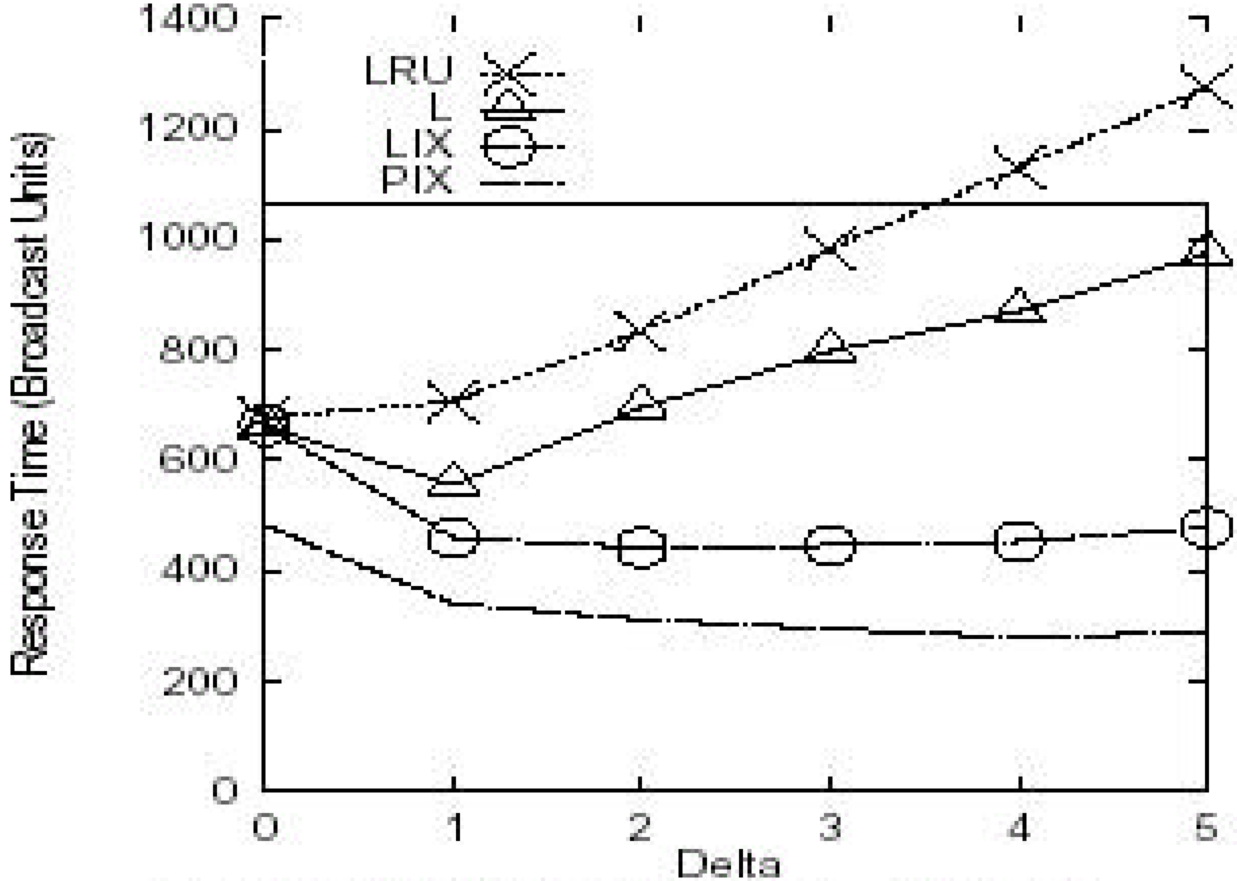
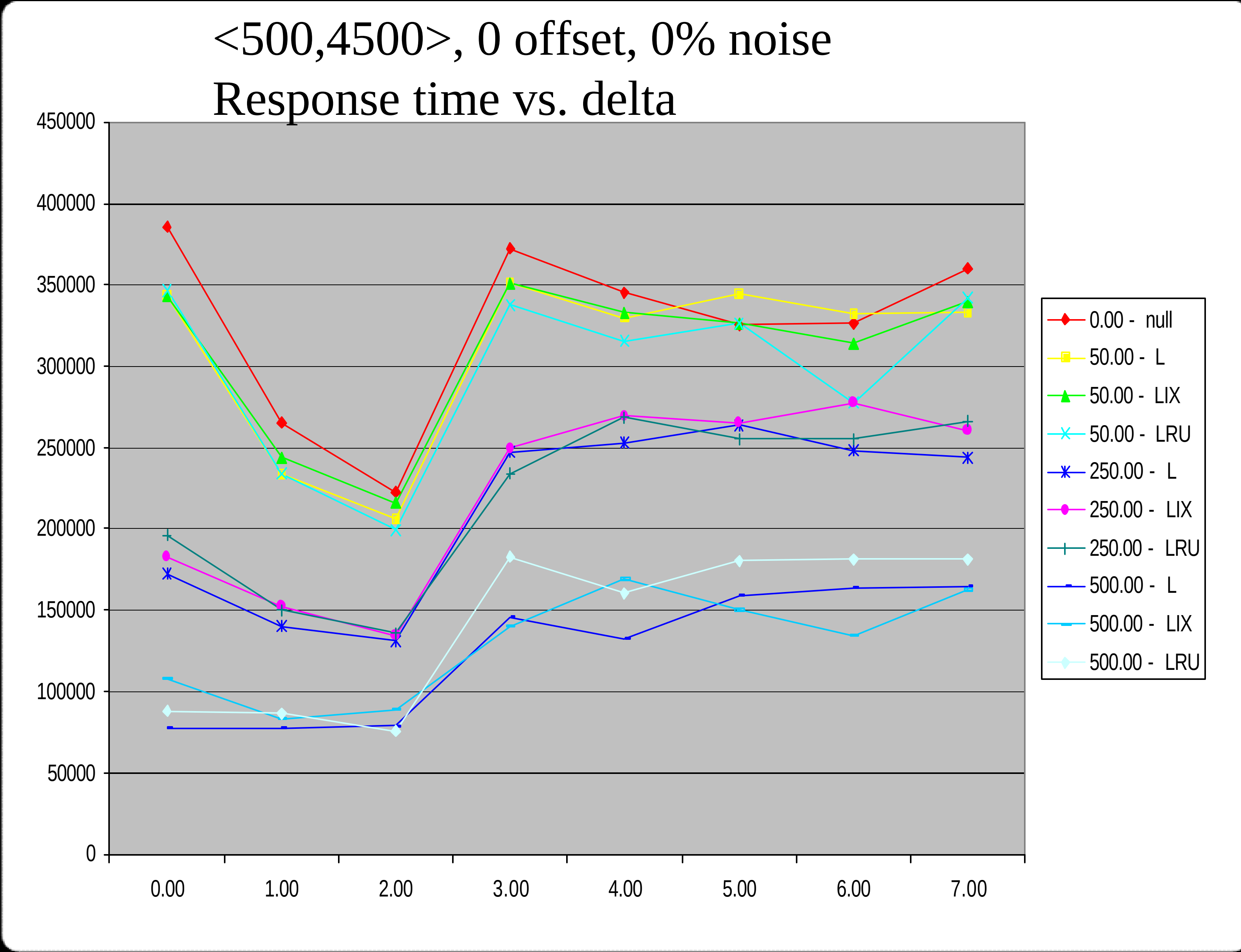


Figure 13: Sensitivity to Δ - Disk D5
CacheSize = 500, Noise = 30%

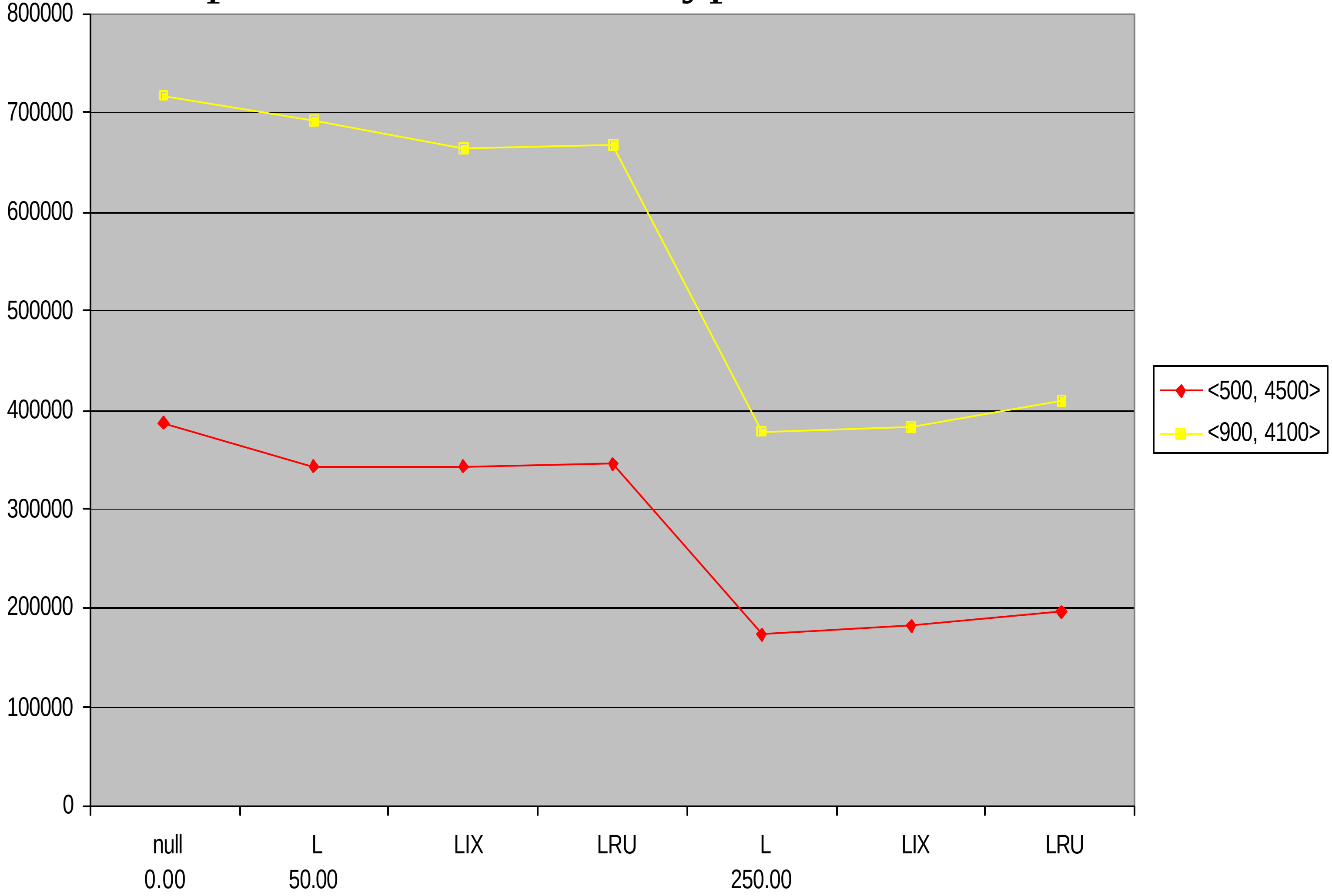
Broadcast Disk Implementation



Broadcast Disk Implementation

0 delta, 0 noise, 0% offset

Response time vs cache types/sizes…. ??????



Broadcast Disk Implementation

- Relative Frequency
  - The speed of a disk, relative to slowest disk.
  - relFreq(i)/relFreq(N) = 1 + ?(N – i)
  - N = the disk speed of the slowest disk.
- Max Chunks
  - max_chunks = LCM (relFreqs)
- NumChunks
  - num_chunks(i): the number of chunks the disk is broken into.
  - num_chunks(i) = max_chunks/rel_freq(i)

Broadcast Disk Implementation

500	4500

Delta = 0
relFreq(1) = 1, relFreq(2) = 1
numChunks(1) = 1, numChunks(2) = 1

Mini cycle 1	0	…	499	500	…	4999

Delta = 1
relFreq(1) = 2, relFreq(2) = 1
numChunks(1) = 1, numChunks(2) = 2

Mini cycle 1	0	…	499	500	…	2749
Mini cycle 2	0	…	499	2750	…	4999

Broadcast Disk Implementation

900	4100

Delta = 0
relFreq(1) = 1, relFreq(2) = 1
numChunks(1) = 1, numChunks(2) = 1

Mini cycle 1	0	…	899	900	…	4999

Delta = 1
relFreq(1) = 2, relFreq(2) = 1
numChunks(1) = 1, numChunks(2) = 2

Mini cycle 1	0	…	899	900	…	2949
Mini cycle 2	0	…	899	2950	…	4999

Broadcast Disk Implementation

- Future Work
  - Run using wireless emulator
  - Implement P and PIX
  - Let tests finish running
  - Implement Server in c or c++
  - Run on dedicated network and machines
  - Run on different packet sizes,
    - current 512 bytes
  - Implement PT and APT caching methods.

Broadcast Disk Implementation

- Conclusion
  - Time limitations and long run times, kept us from gathering all our results
  - Still L and LIX benefited from larger caches and larger deltas.
  - At small caches overhead of L and LIX not really worthwhile.