ROCCSS: A Real-time Online Coded Caching Simulator

Lecturer **Yuxin Tang**

Date **2 Jun, 2017**

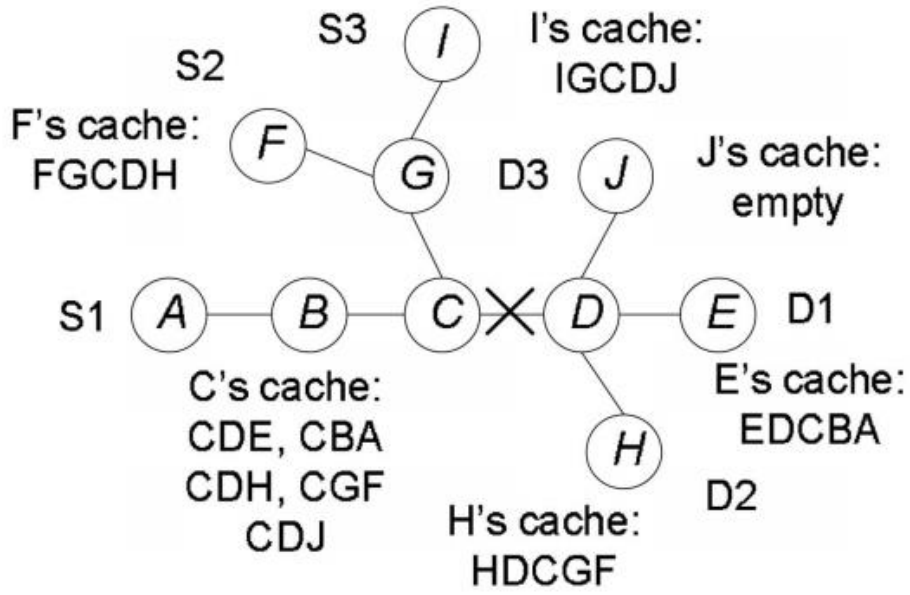
Department of Computer Science and Engineering

Shanghai Jiao Tong University, P.R.China

Outline

Outline

1.1 Related Work



An Example of Routing Caching in DSR

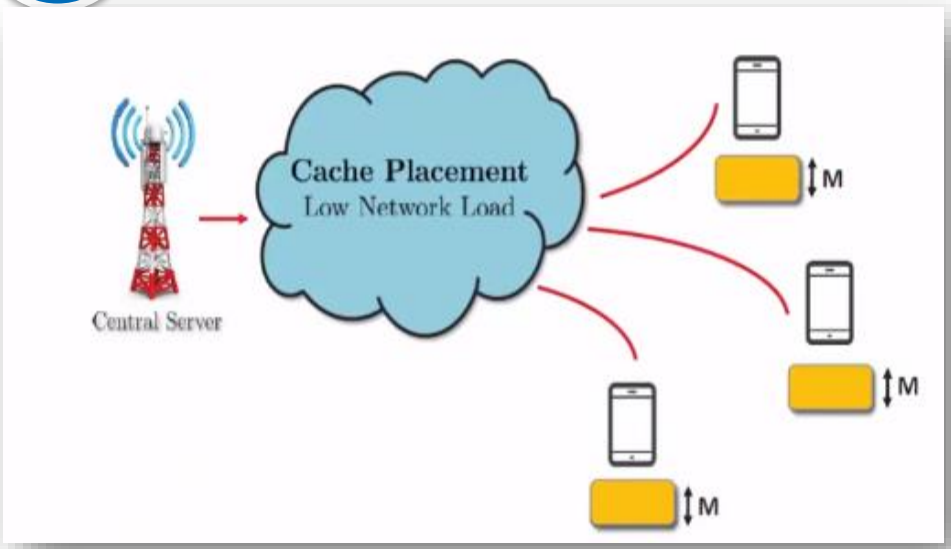
Caching Updating Problem

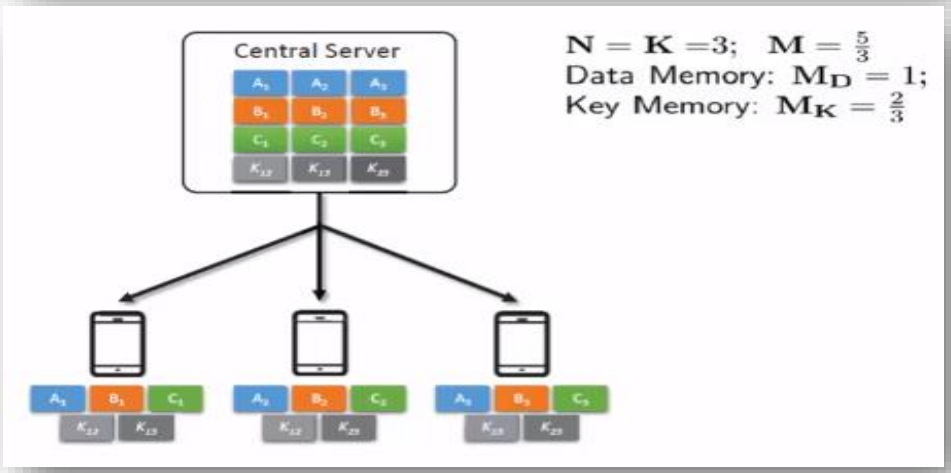
- File recovery
- Updating algorithm
- Speed & Energy

Coded Caching

- Multicasting opportunities
- Minimize delivery rate

1.1 Related Work





Coded Caching

M. A. Maddah-Ali and U. Niesen, Fundamental limits of caching,arXiv:1209.5807 [cs.IT], Sept. 2012. Submitted to IEEE Trans. Inf.Theory.

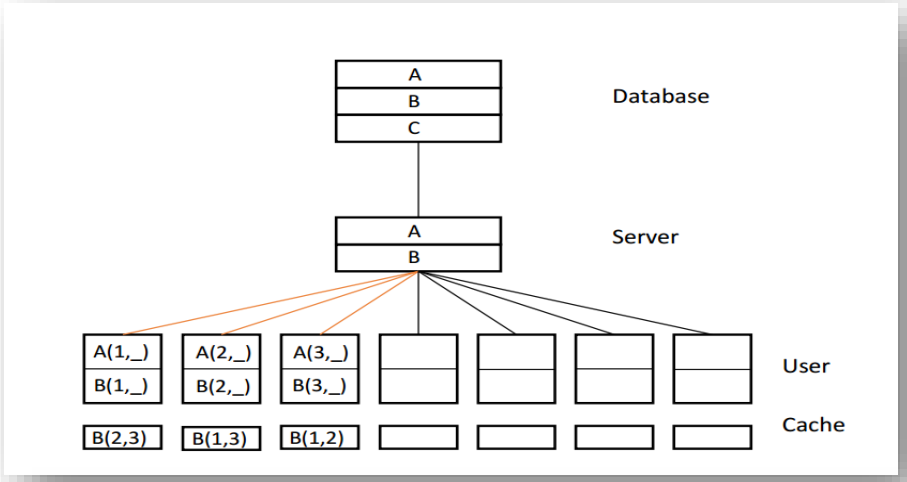
M. A. Maddah-Ali and U. Niesen, Decentralized coded caching attains order-optimal memory-rate tradeoff, arXiv:1301.5848 [cs.IT], Jan. 2013

U. Niesen and M. A. Maddah-Ali, Coded caching with nonuniform demands, arXiv:1308.0178 [cs.IT], Aug. 2013.

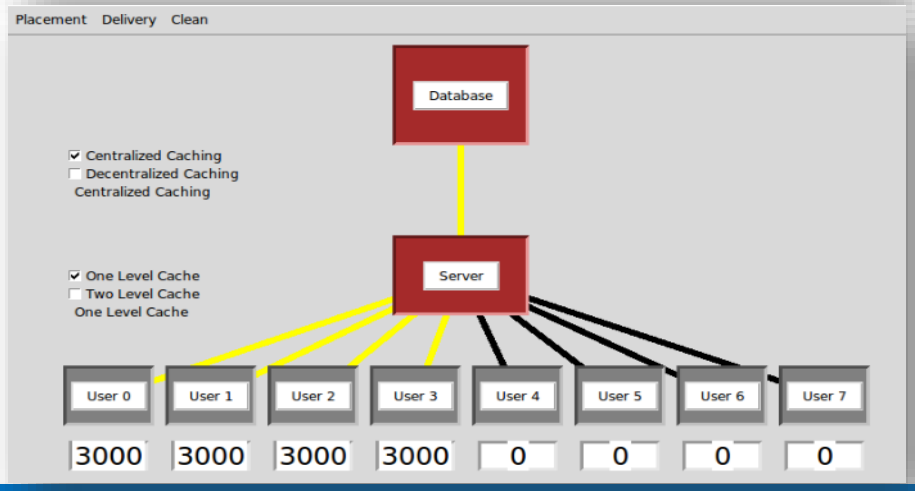
Outline

1.2 Our Contribution

Our Design:



- Video Streaming Transmission
- Multi-Delivery Mode Cache Pattern
- Programmable Module



- Real-Time Analysis
- Extendable Simulation
- Customized System Parameters

Outline

2.1 Problem Setting

Controller Algorithm: Decentralized Coded Caching

Algorithm A1 Decentralized Coded Caching

1: **procedure** PLACEMENT ▷ The placement phase
2: **for** $k \in [K], n \in [N]$ **do**
3: User k caches a random $\frac{MF}{N}$ bit subset of file n
4: **end for**
5: **end procedure**
6: **procedure** DELIVERY$(d_1, ..., d_k)$ ▷ The delivery phase
7: **for** $s = K, K-1, ..., 1$ **do**
8: **for** $S \subset [K] : [S] = s$ **do**
9: Server sends $\oplus_{k \in S} V_{k,S\backslash\{k\}}$
10: **end for**
11: **end for**
12: **end procedure**
13: **procedure** DELIVERY'$(d_1, ..., d_k)$ ▷ The delivery phase
14: **for** $n \in [N]$ **do**
15: Server sends enough random linear combinations of bits in file n for all users requesting it to decode
16: **end for**
17: **end procedure**

Remark 1: The $\oplus$ in Line 9 of Algorithm 1 represents the bit-wise XOR operation. All elements $V_{k,S\backslash\{k\}}$ are assumed to be zero padded to the length of the longest element.

- A. Maddah-Ali and U. Niesen, Decentralized coded caching attains order-optimal memory-rate tradeoff, arXiv:1301.5848 [cs.IT], Jan. 2013

- User can prefetch data randomly during placement

2.1 Problem Setting

Goal: Minimize Delivery Rate

$$\bar{R} \triangleq \lim_{t\to\infty} sup \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{E}(R_t)$$

$$R(M,N,K,D) \triangleq K(1-\frac{M}{N})\frac{N}{KM}(1-(1-\frac{M}{N})^K)$$

Scheduling

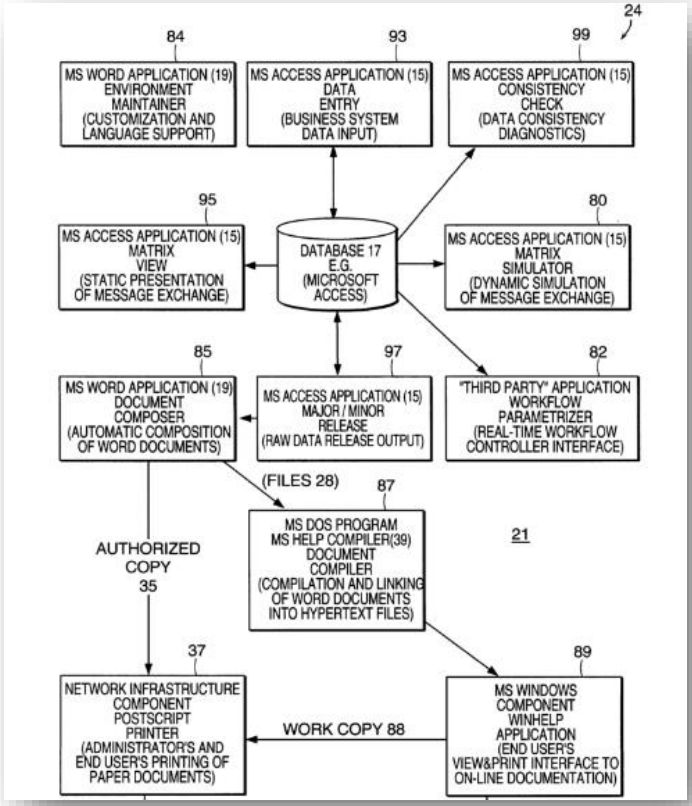
- Least Recently Used(LRU)
- Most Recently Used(MRU)
- Pseudo LRU(PLRU)
- First In First Out(FIFO)

Outline

2.2 System Overview

Database:

- Rules
  - With $p, N_t = N_{t-1}$
  - With $1 - p$, a file will be evicted
- Design
  - Relational DB
  - Equal Size Storage
  - File Recovery



2.2 System Overview

Server:

Algorithm A2 Chunk Dividing Algorithm

1: **procedure** DIVIDE ▷ Dividing Process
2: seed()
3: **for** $n \in [N]$ **do**
4: Calculate $RandomSize = Random()$
5: Calculate $t = \frac{KM}{N}$ and $q = \frac{F}{combination(K,t)}$
6: Split files according to $argmax(F(x) = \frac{log(t)*RandomSize}{q*q^t})$
7: **end for**
8: **end procedure**

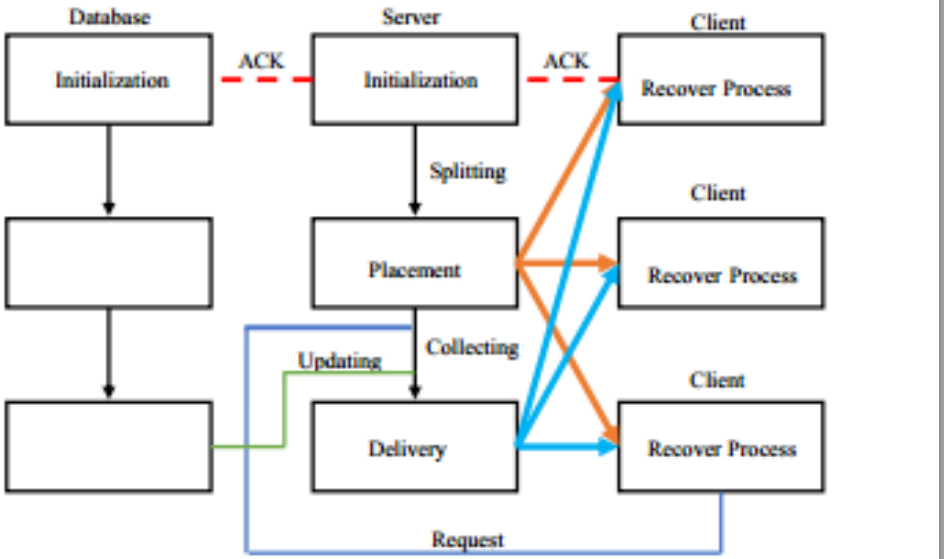
Chunk Dividing Algorithm

Module in Server

- Chunk Dividing Module
- Delivery Module
- Placement Module

2.2 System Overview

Client:



Functions

- Multi-level Cache Option
- Cache Combination
- File Recovery

Signal

- ACK
- Splitting & Collecting
- Updating & Request

Outline

3.1 Experiment

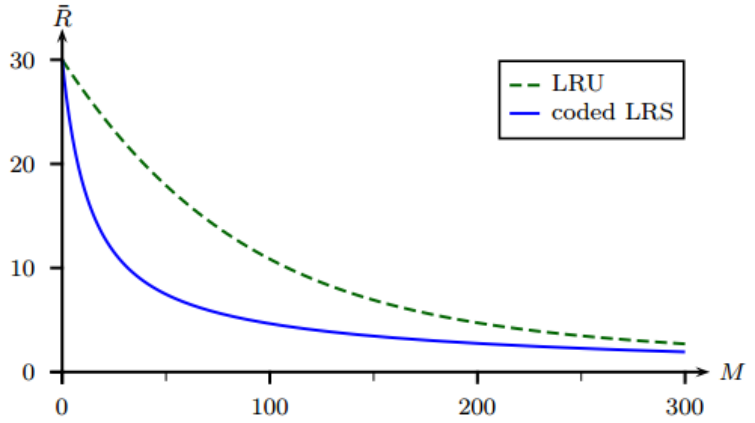
Theoretical Results :

- Main theoretical results
  - Upper Bound

$$\frac{1}{12}R(M,N,K) \le \bar{R}^{\star} \le 2R(M,N,K)+6$$

  - Lower Bound

$$\bar{R}^{\star} \ge \frac{1}{12}R(M,N,K)$$



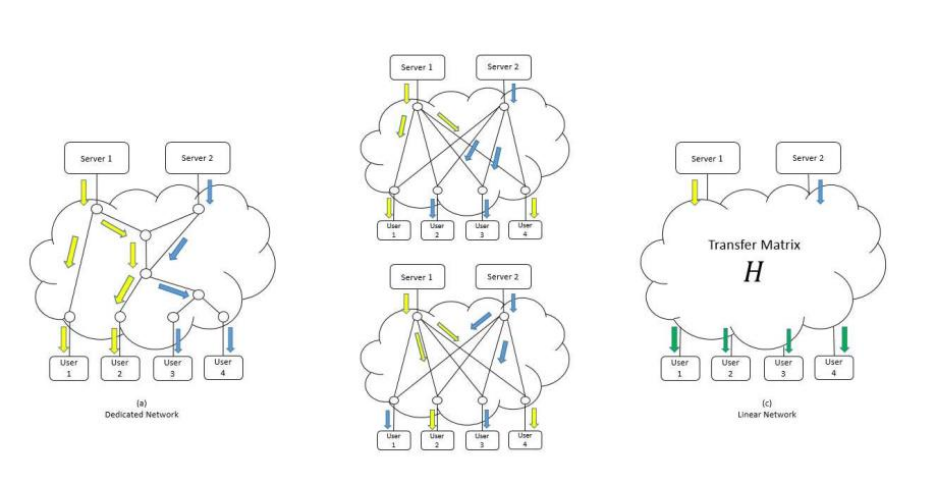
$$evict_{M}^{FIFO}(k) = fill_{M}^{FIFO}(k) = k$$

$$fill_{HM}^{MRU}(k) = fill_{M}^{MRU}(k) = \infty$$

- No similar property in previous designs
- Good performance due to high speed of transmission

3.1 Experiment

Efficiency :



- O(N^2log(N)) for delivery
- In previous work: O(2^N) for the whole process

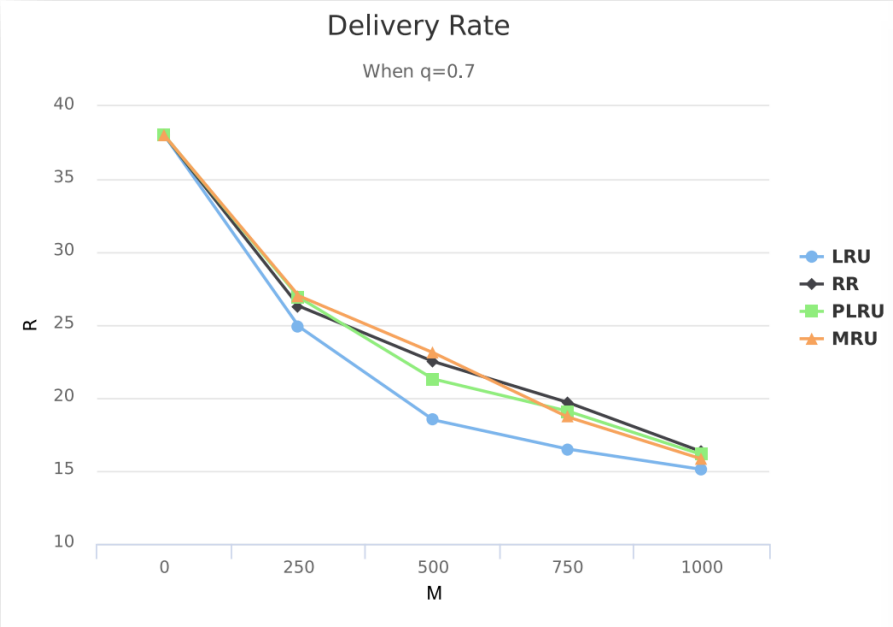
Outline

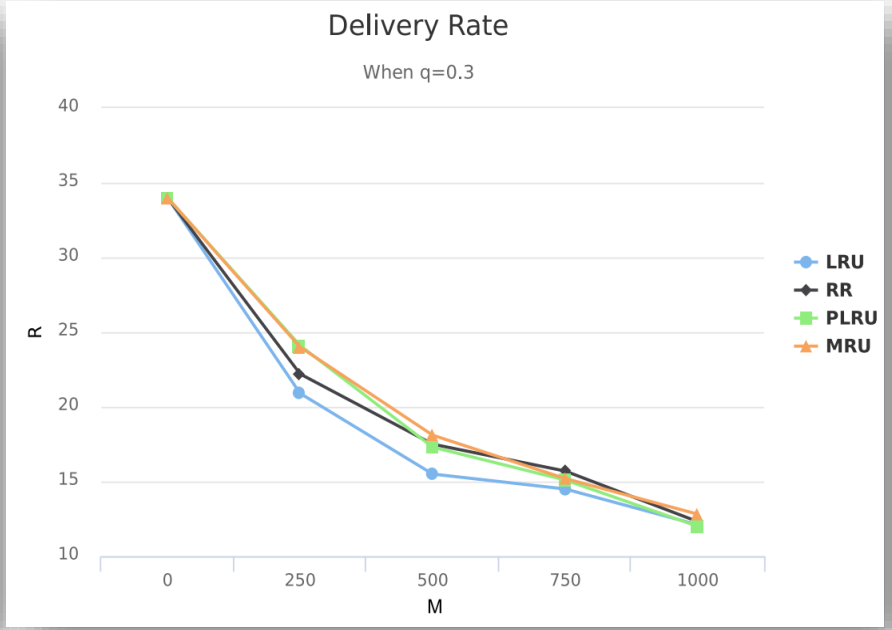
3.2 Result

Experiment 1 : Q-value One Level Cache

3.2 Result

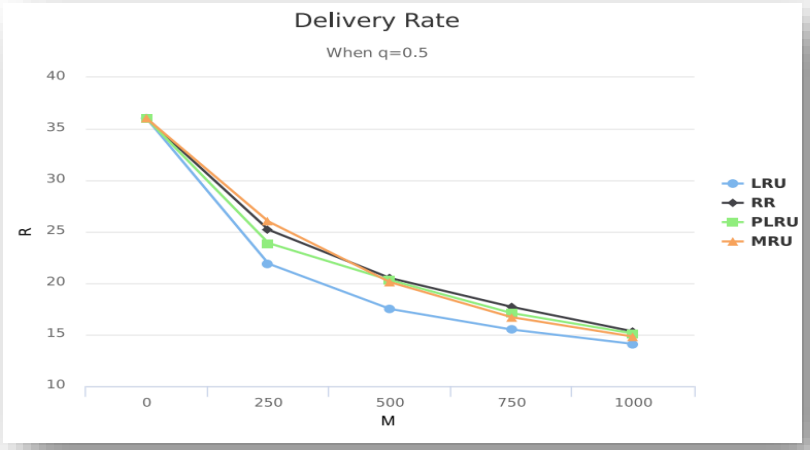
Experiment 2: Q-value Two Level Cache

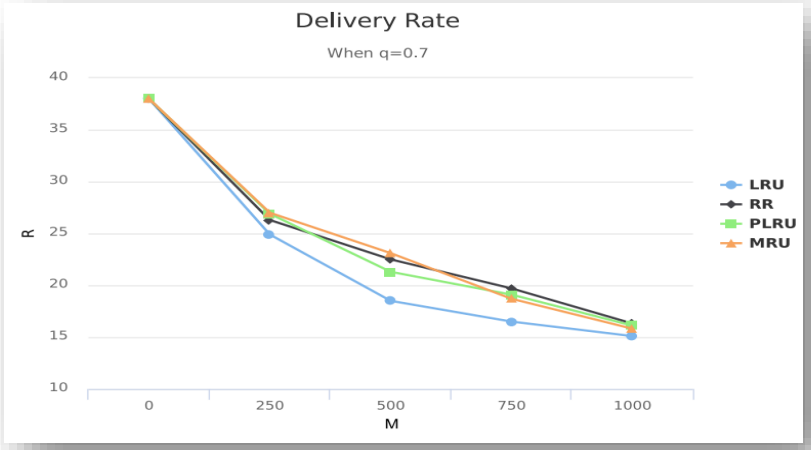


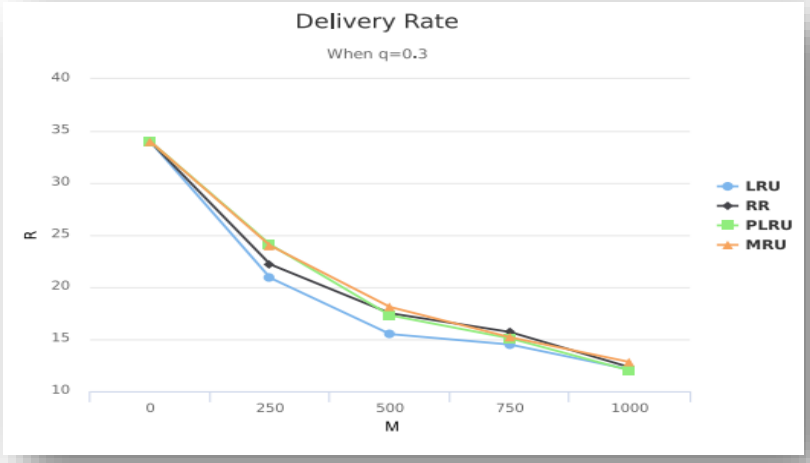


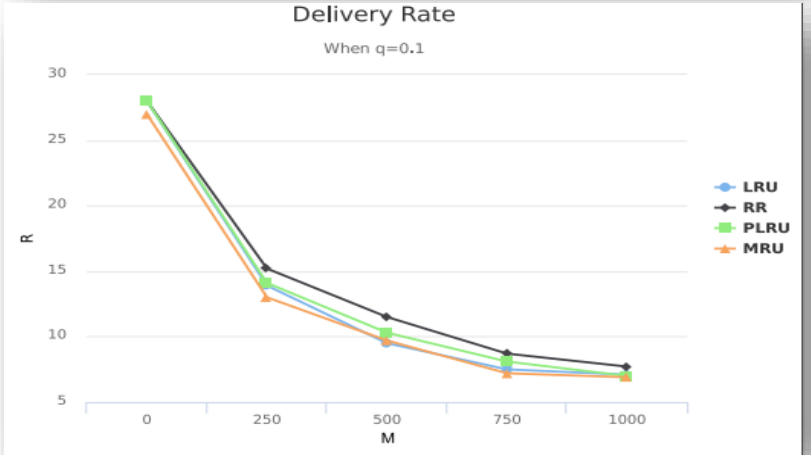
3.2 Result

Experiment 3: Different Q-value



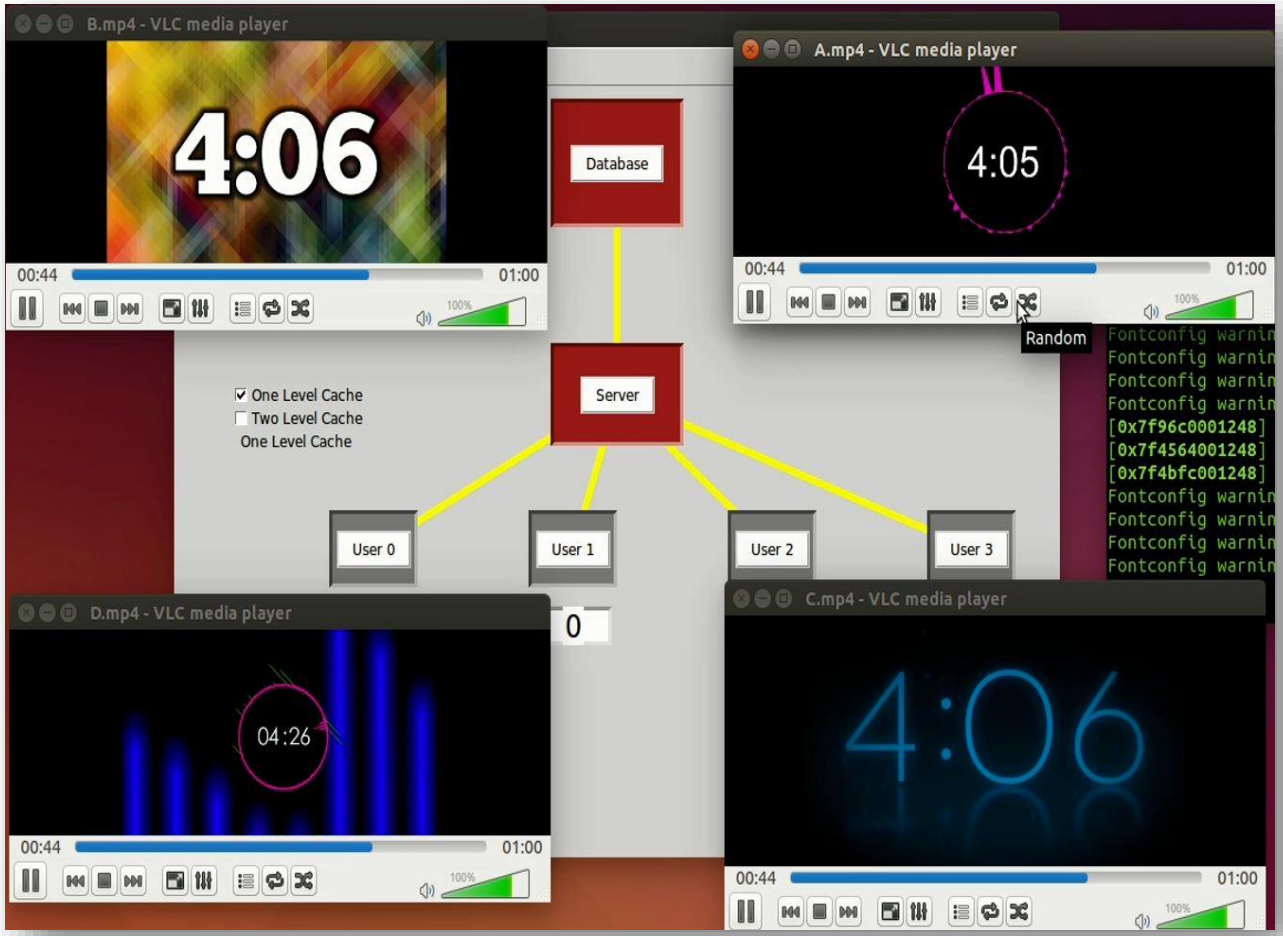




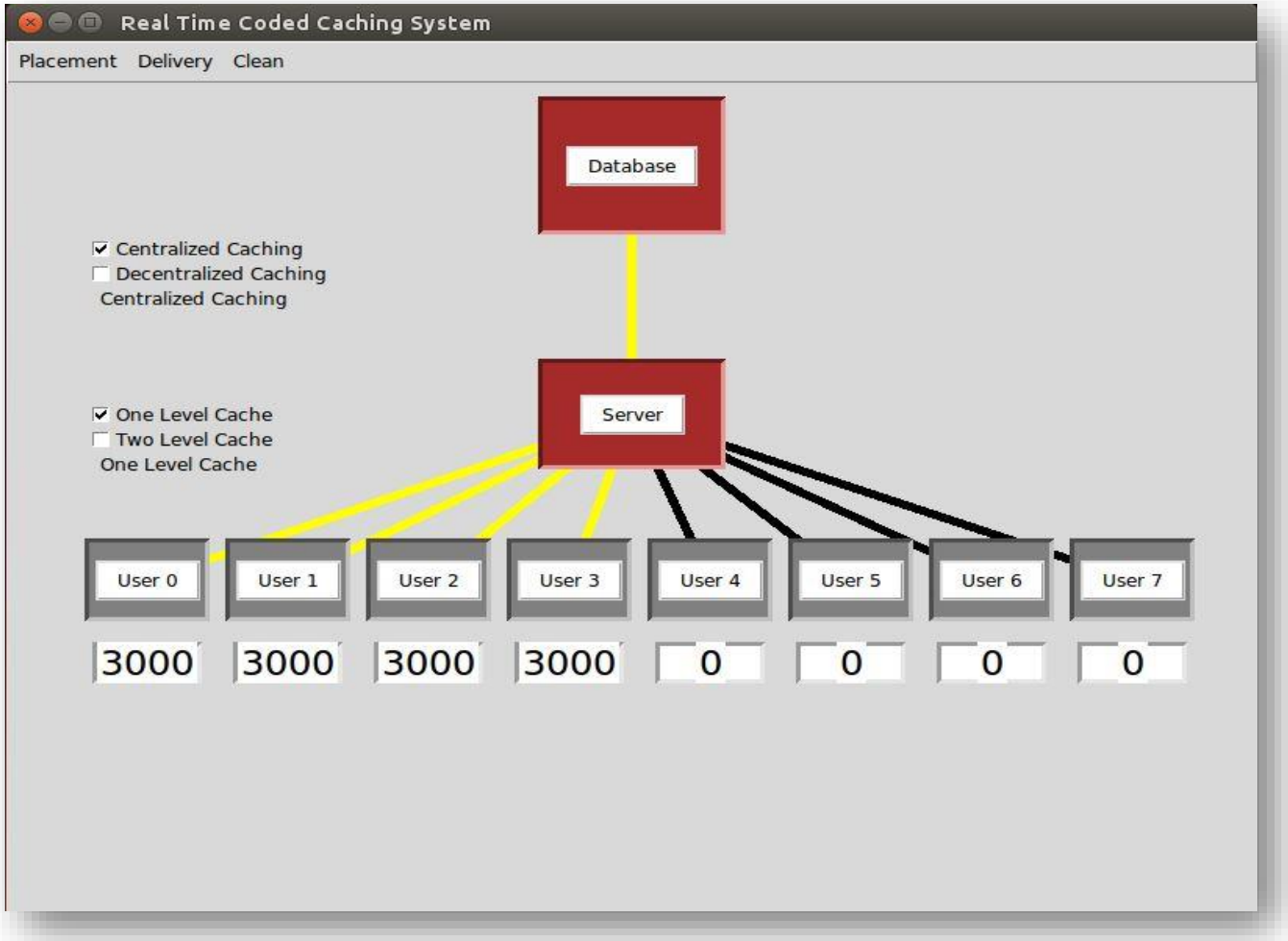


Outline

4 Demo


B.mp4 - VLC media player
4:06
00:44
01:00
A.mp4 - VLC media player
4:05
00:44
01:00
Random
Database
Server
One Level Cache
Two Level Cache
One Level Cache
User 0
User 1
User 2
User 3
0
D.mp4 - VLC media player
04:26
00:44
01:00
C.mp4 - VLC media player
4:06
00:44
01:00
Fontconfig warnin
[0x7f96c0001248]
[0x7f4564001248]
[0x7f4bfc001248]

4 Demo


Real Time Coded Caching System
Placement
Delivery
Clean
Database
Centralized Caching
Decentralized Caching
Centralized Caching
Server
One Level Cache
Two Level Cache
One Level Cache
User 0
User 1
User 2
User 3
User 4
User 5
User 6
User 7
3000
3000
3000
3000
0
0
0
0


Thanks!
Q&A