Caching: A Feedback Perspective

Mohammad Ali Maddah-Ali
Bell Labs, Alcatel-Lucent

joint work with

Urs Niesen

Communication Theory Workshop
June 2013

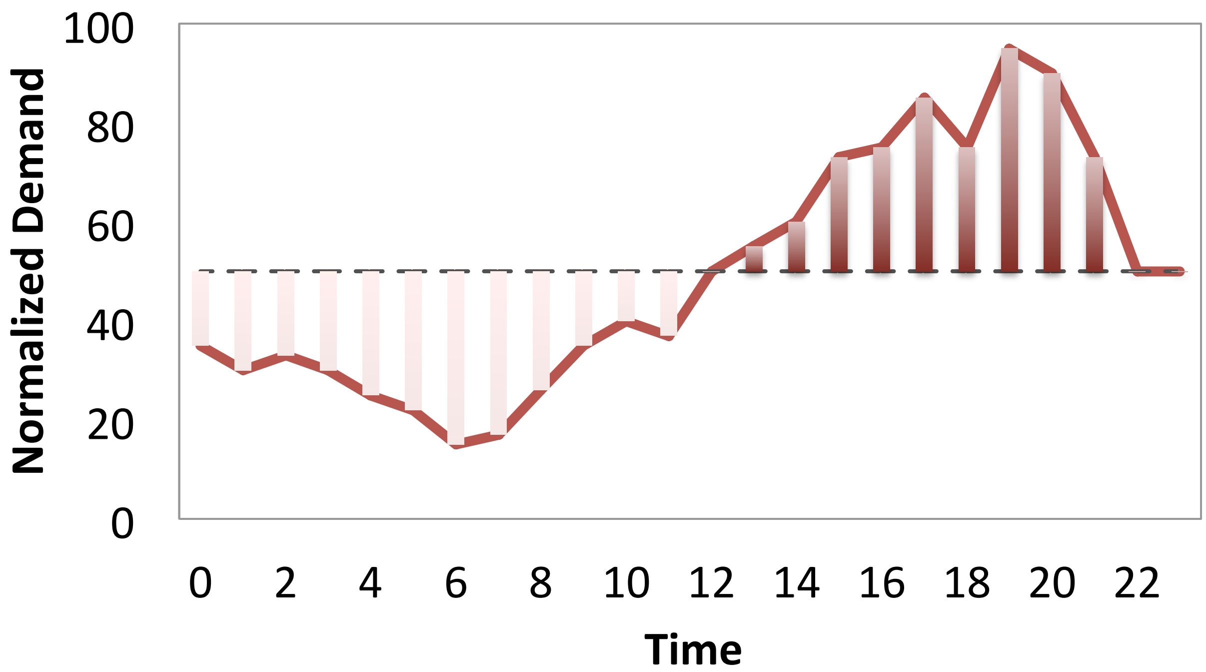
Video on Demand

- Video on Demand is getting increasingly popular
  - Netflix Streaming Service
  - Amazon Instant Video
  - Hulu
  - Verizon/Comcast on Demand
  - …

Place Significant Stress on Service Providers Network.

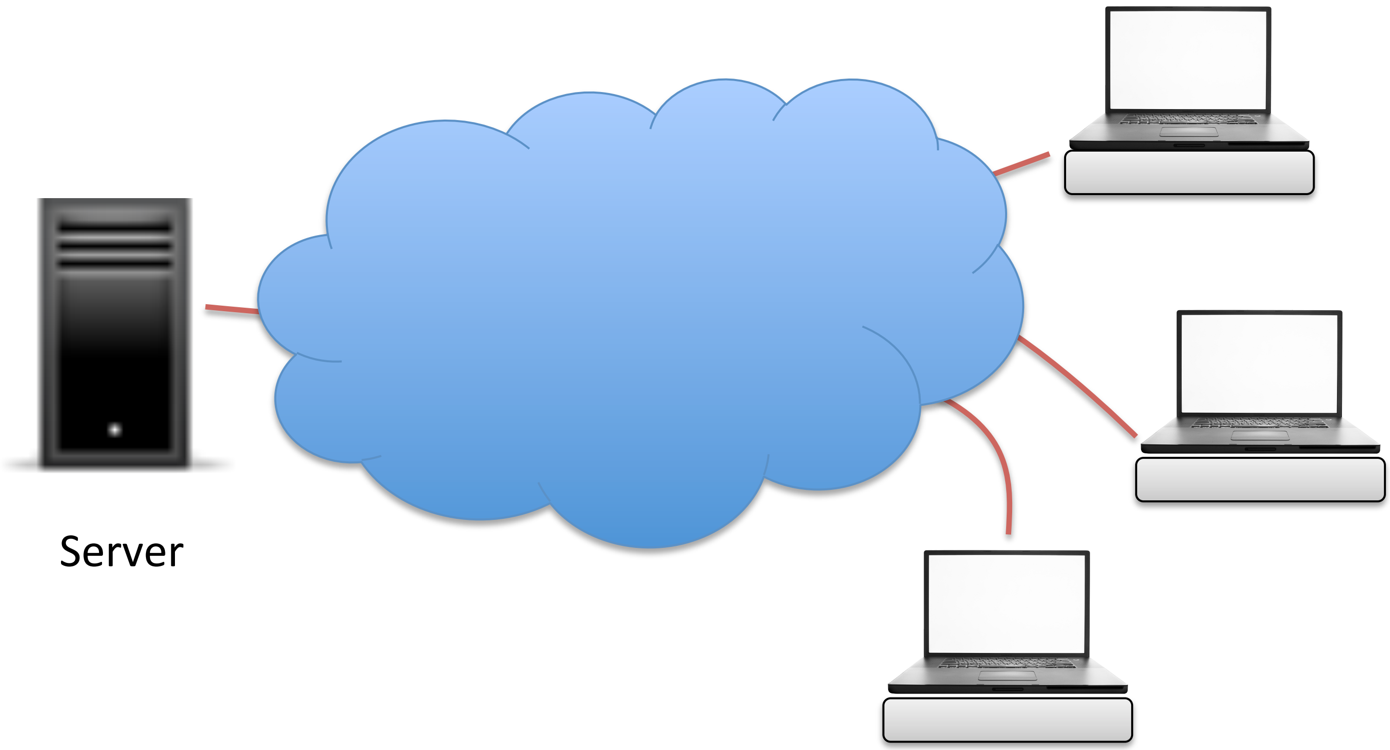
Prefetching can be used to mitigate this stress.

Temporal Behavior



- High temporal traffic variability
- Caching (Prefetching) can help to smooth traffic

Caching (Prefetching)

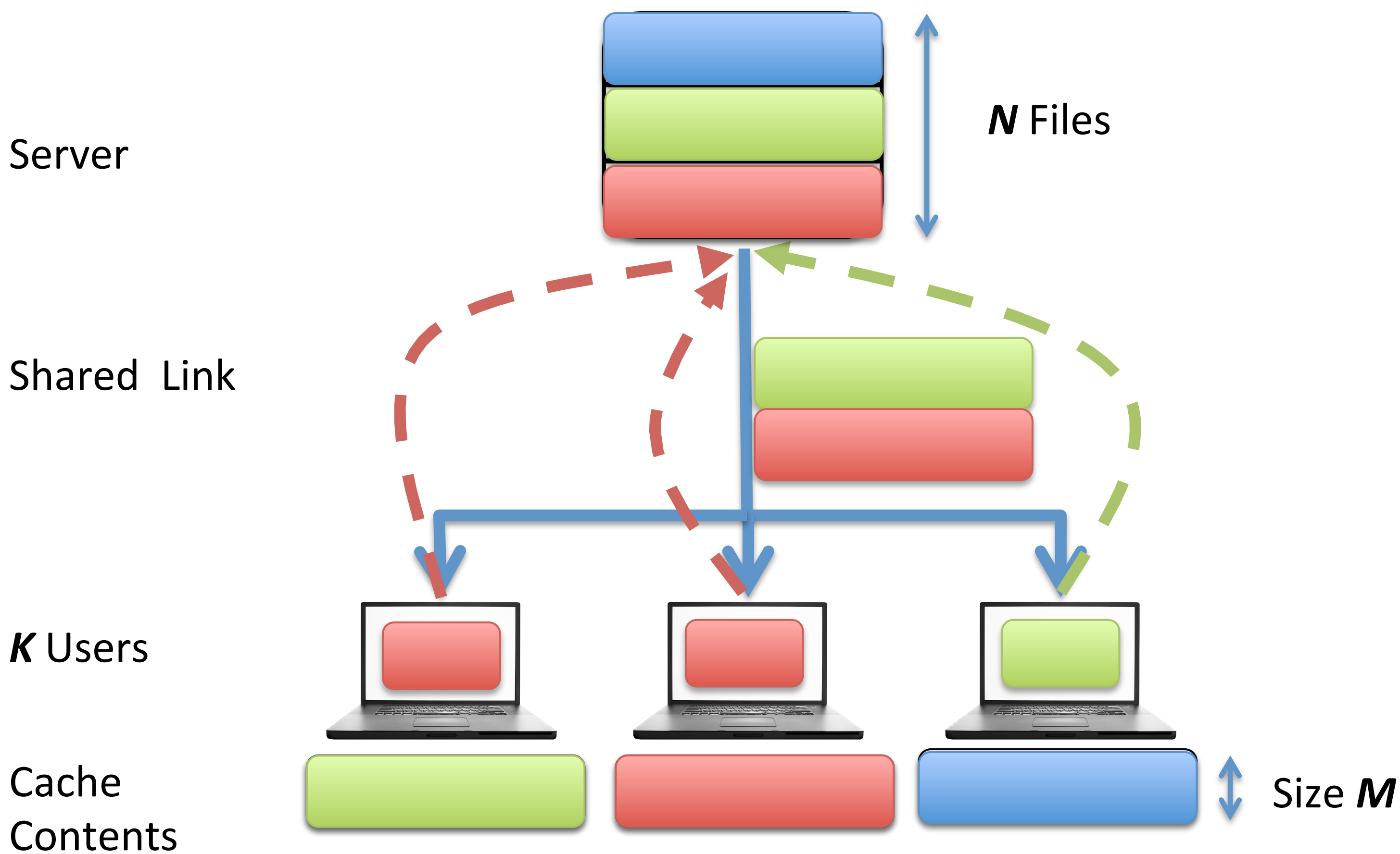


- **Placement Phase:** Populate caches
- **Delivery Phase:** Deliver Content

What Should We Cache?

- Early feedback (demands) from users
  - Demands known BEFORE prefetching
  - Cache the requested demand in nearby memory
  - Role of Cache: To deliver part of data locally.
- Late feedback from users (instantaneous demand)
  - Demands are known AFTER prefetching
  - What Should be cached?
  - What is the role of caching?

Problem Setting



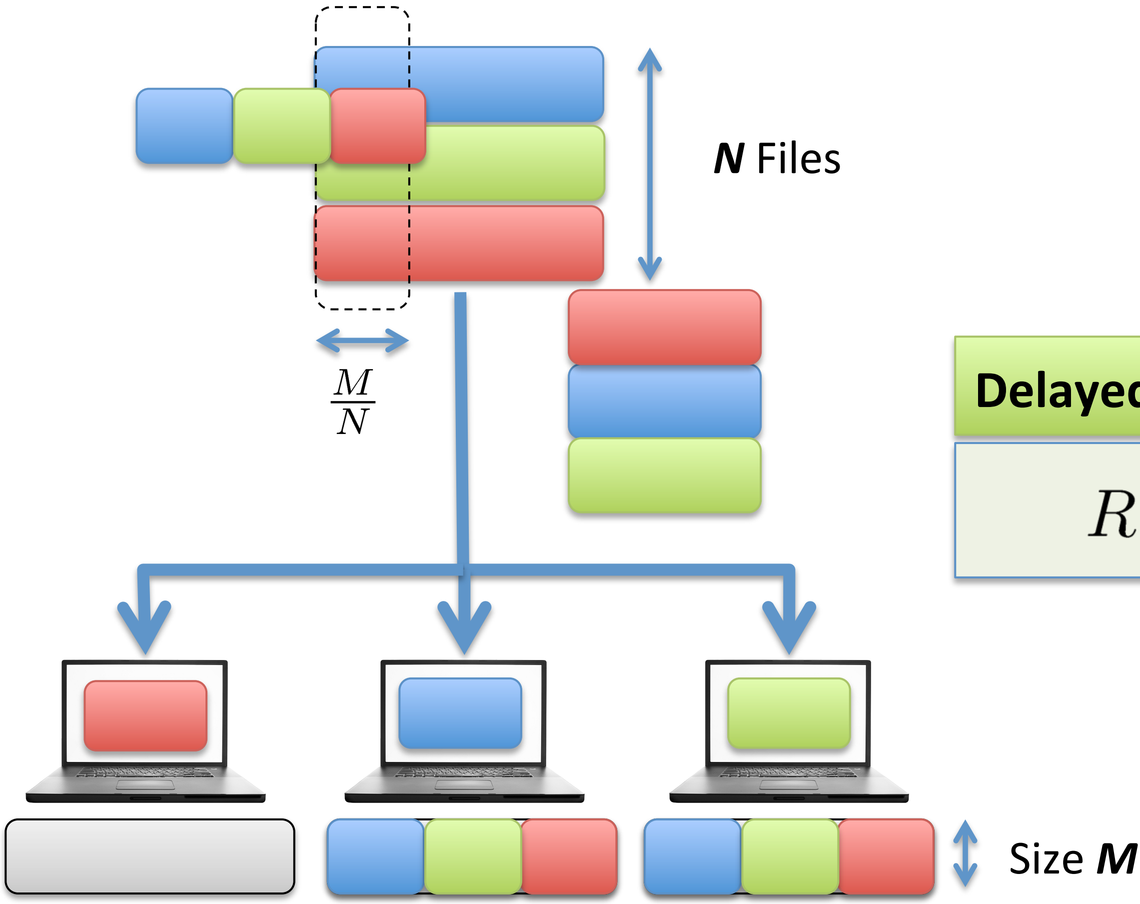
Placement: Cache arbitrary functions of the files (linear, nonlinear, …)

Delivery: Requests are revealed to the server
- Server sends a function of the files

Question: Smallest worst-case rate R(M) needed in delivery phase?

How to choose (1) caching functions, (2) delivery functions

Conventional Caching



Delayed Feedback (Conventional)

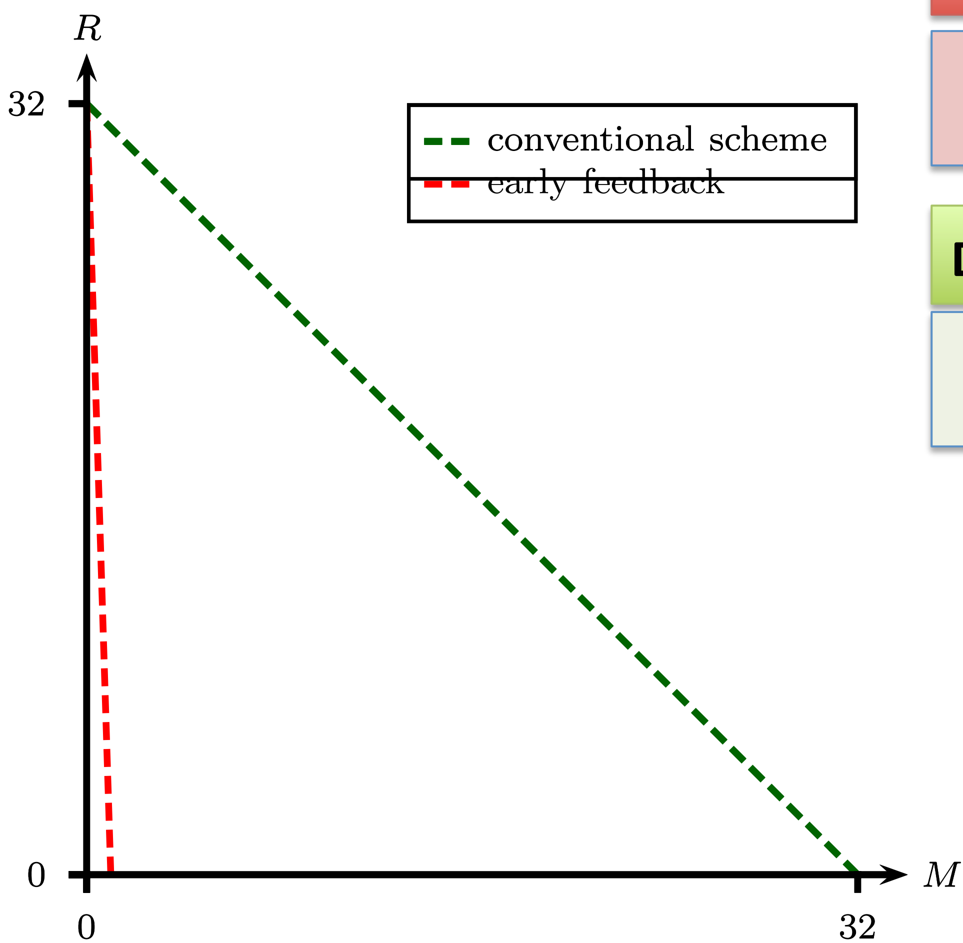
$$R(M) = K(1 - \frac{M}{N})$$

Gain of Caching: Function (normalized) local cache size

Basic Role of Caching: Part of the file is delivered locally

Comparison

N Files, K Users, Cache Size M



Rate (Early Feedback)

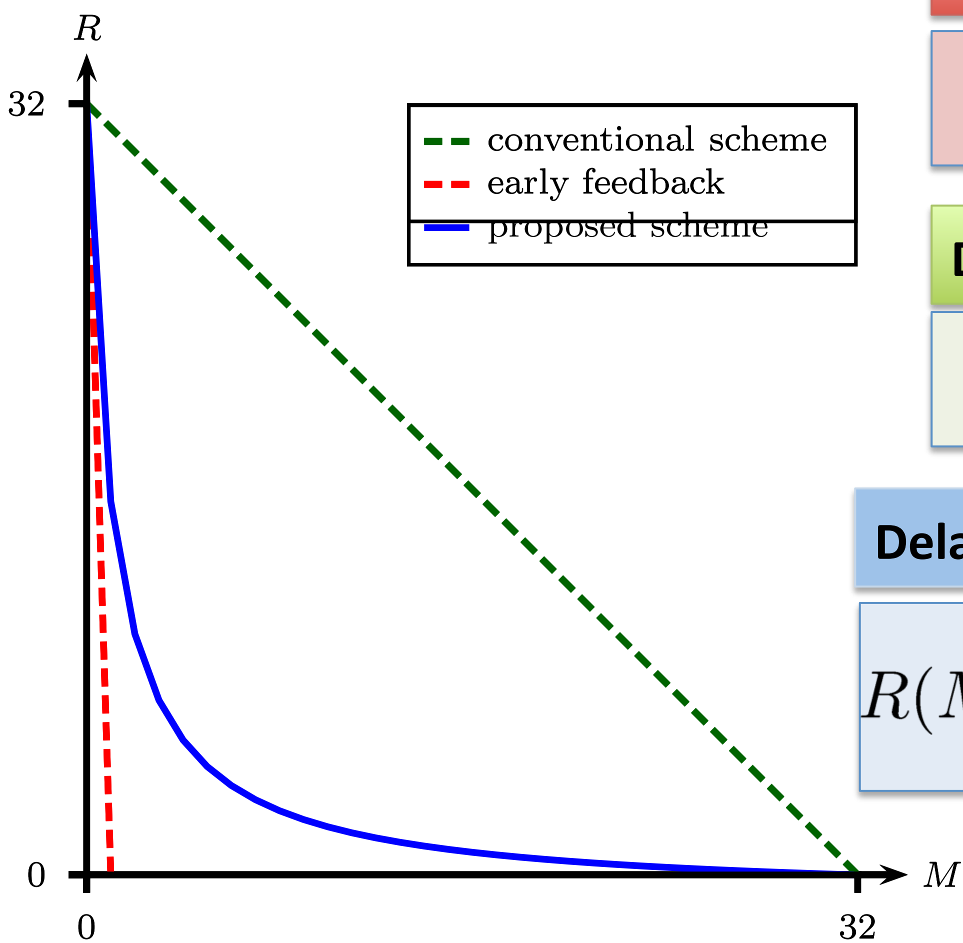
$$R(M) = K(1 - M)$$

Delayed Feedback (Conventional)

$$R(M) = K(1 - \frac{M}{N})$$

Comparison

N Files, K Users, Cache Size M



Rate (Early Feedback)

$$R(M) = K(1 - M)$$

Delayed Feedback (Conventional)

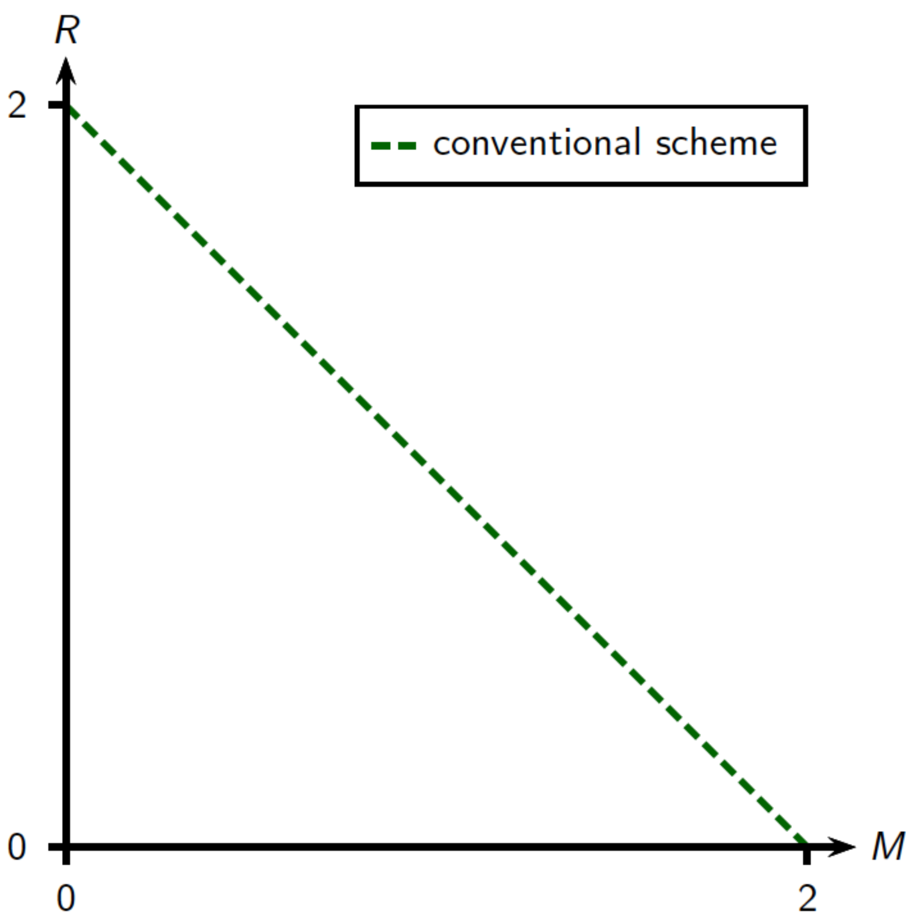
$$R(M) = K(1 - \frac{M}{N})$$

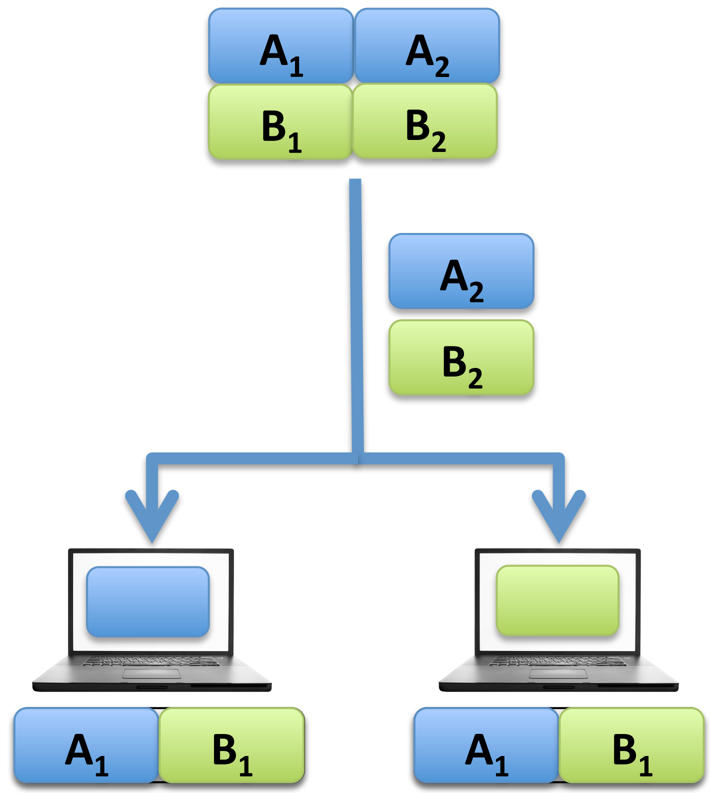
Delayed Feedback (Proposed)

$$R(M) = K(1 - \frac{M}{N})\frac{1}{1+KM/N}$$

Conventional Scheme (Recall)

N=2 Files, K=2 Users, Cache Size M=1

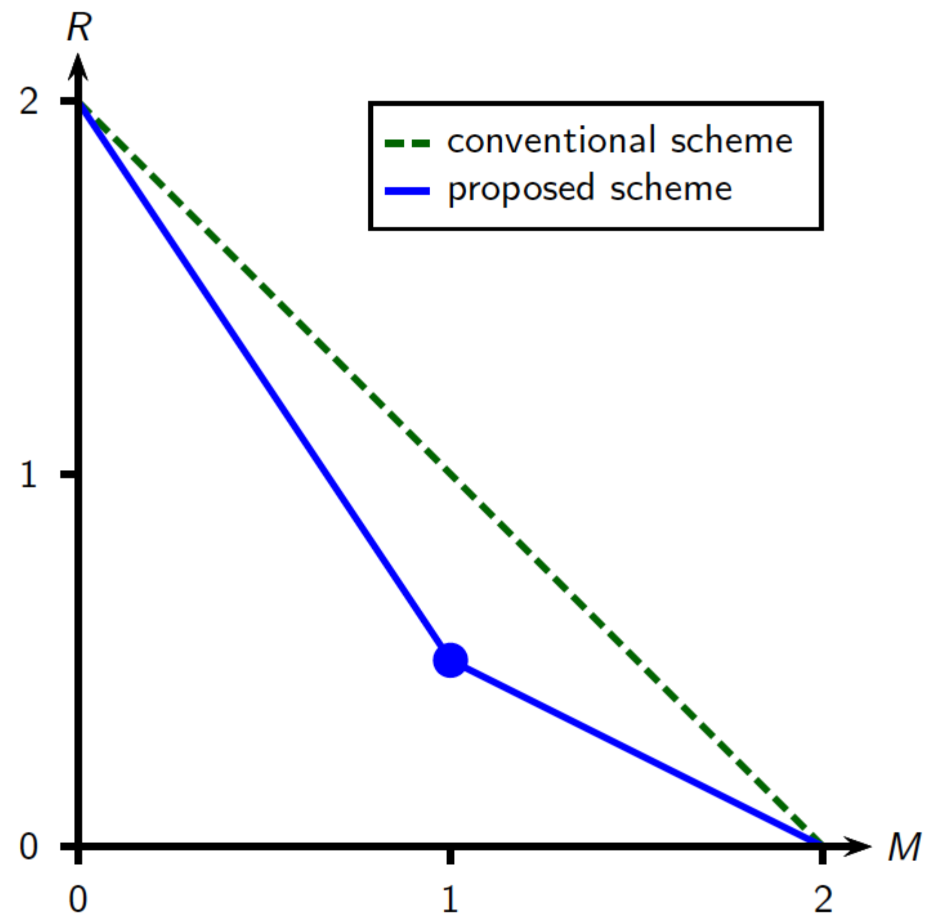


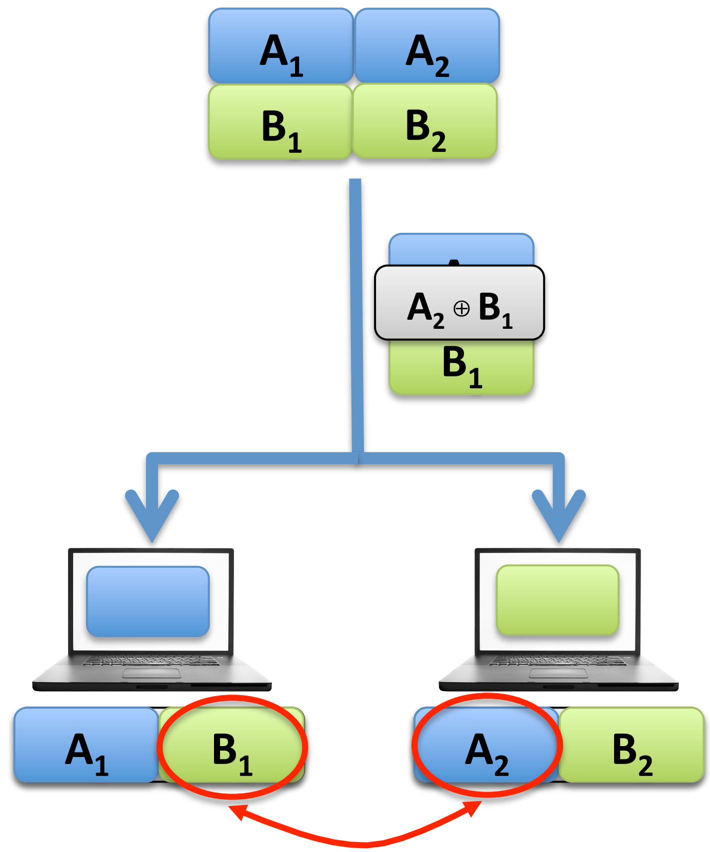


Multicasting opportunity only possible for users with the **same** demand

Proposed Scheme

N=2 Files, K=2 Users, Cache Size M=1

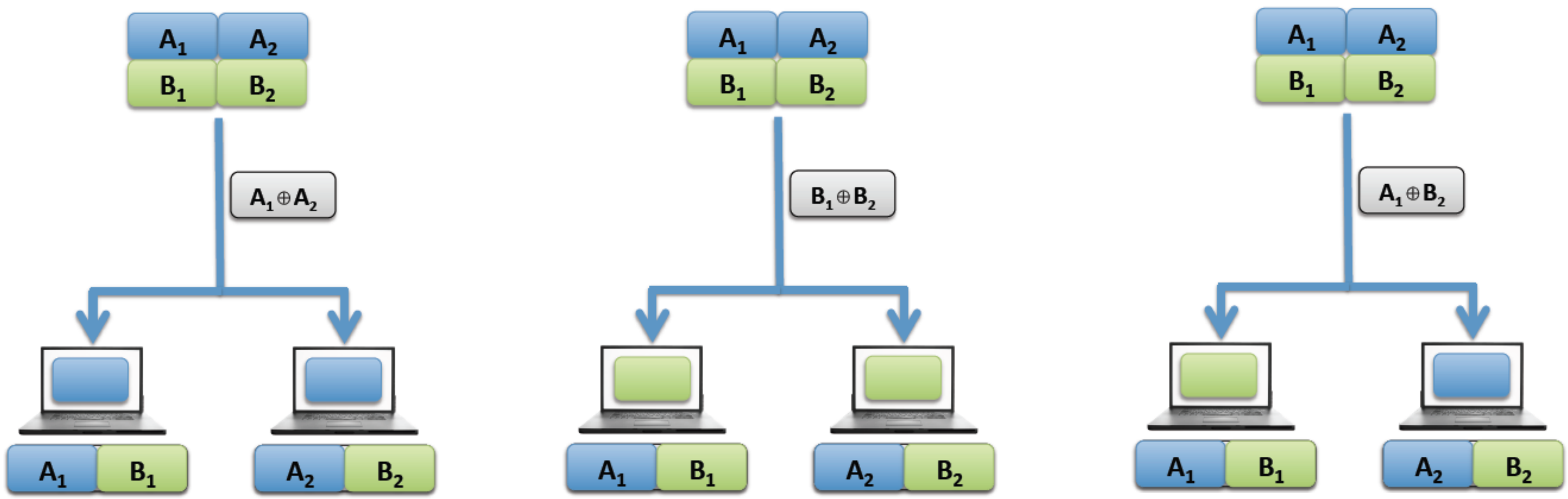




Multicasting opportunity for users with **different** demand

Proposed Scheme

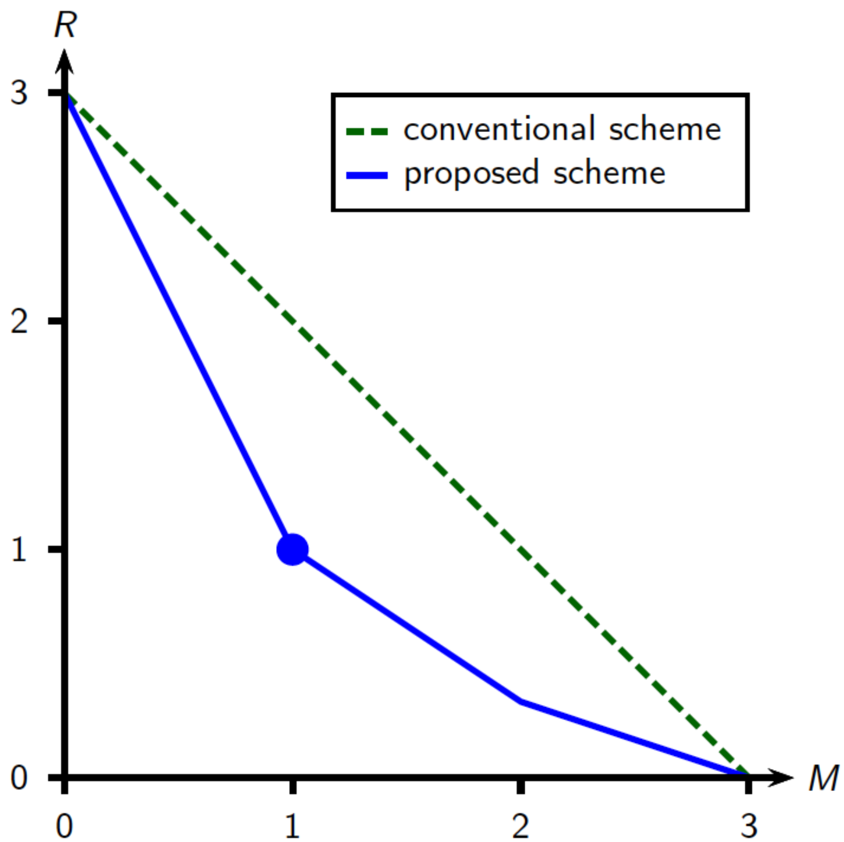
N=2 Files, K=2 Users, Cache Size M=1

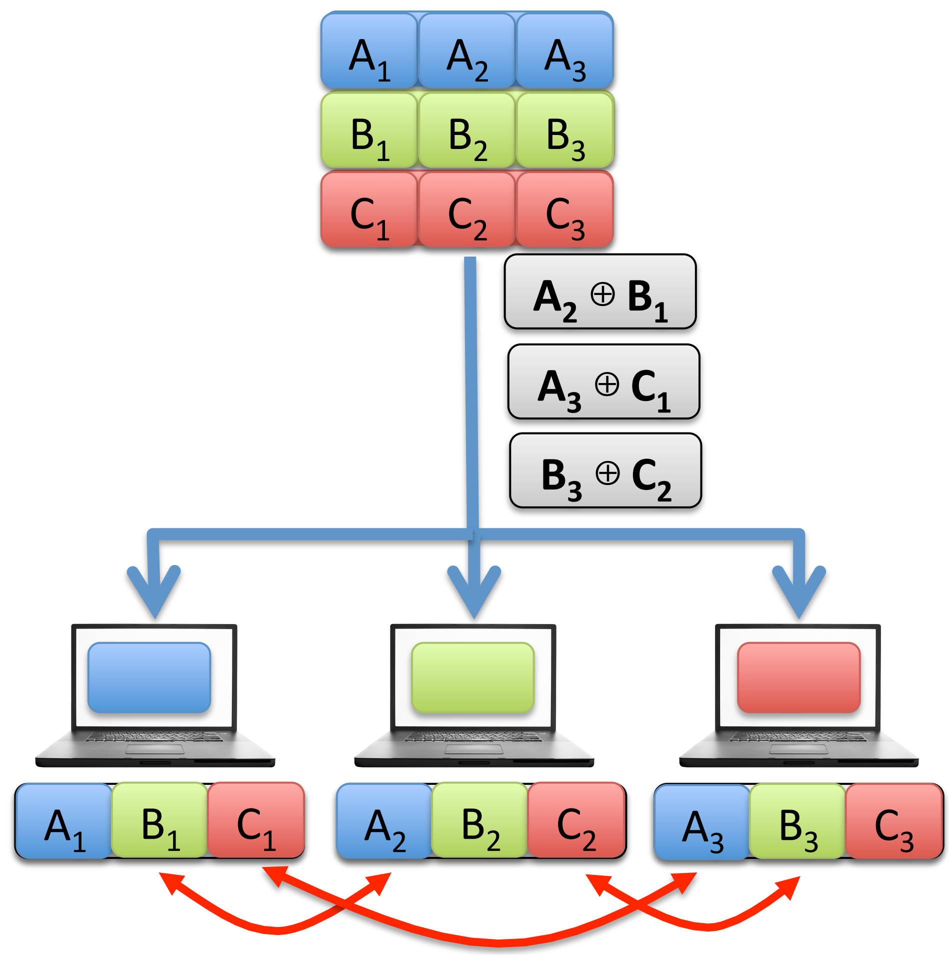


Simultaneous Multicasting Opportunity

Proposed Scheme

N=3 Files, K=3 Users, Cache Size M=1

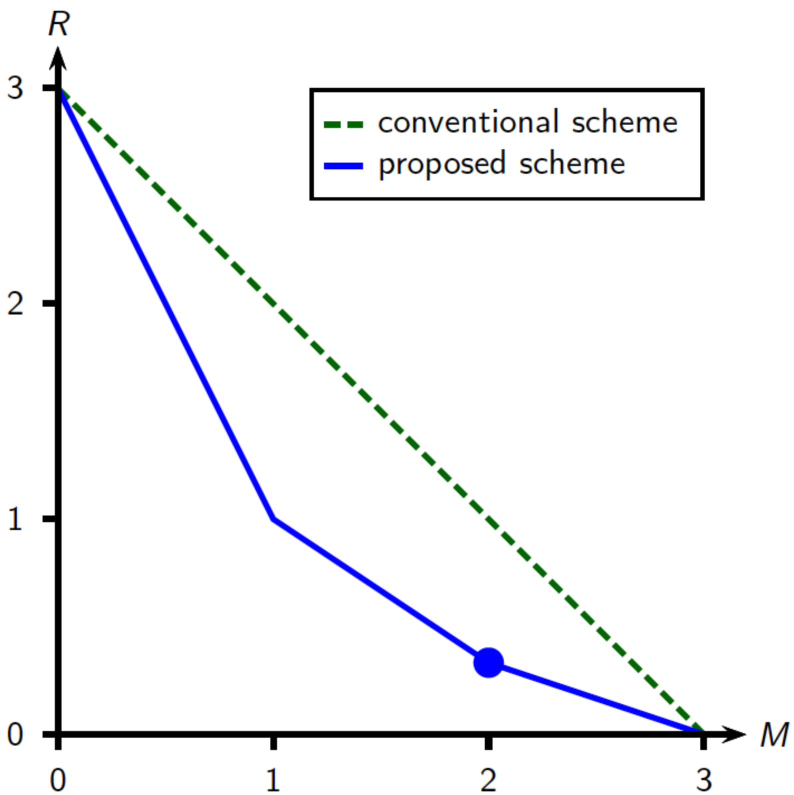


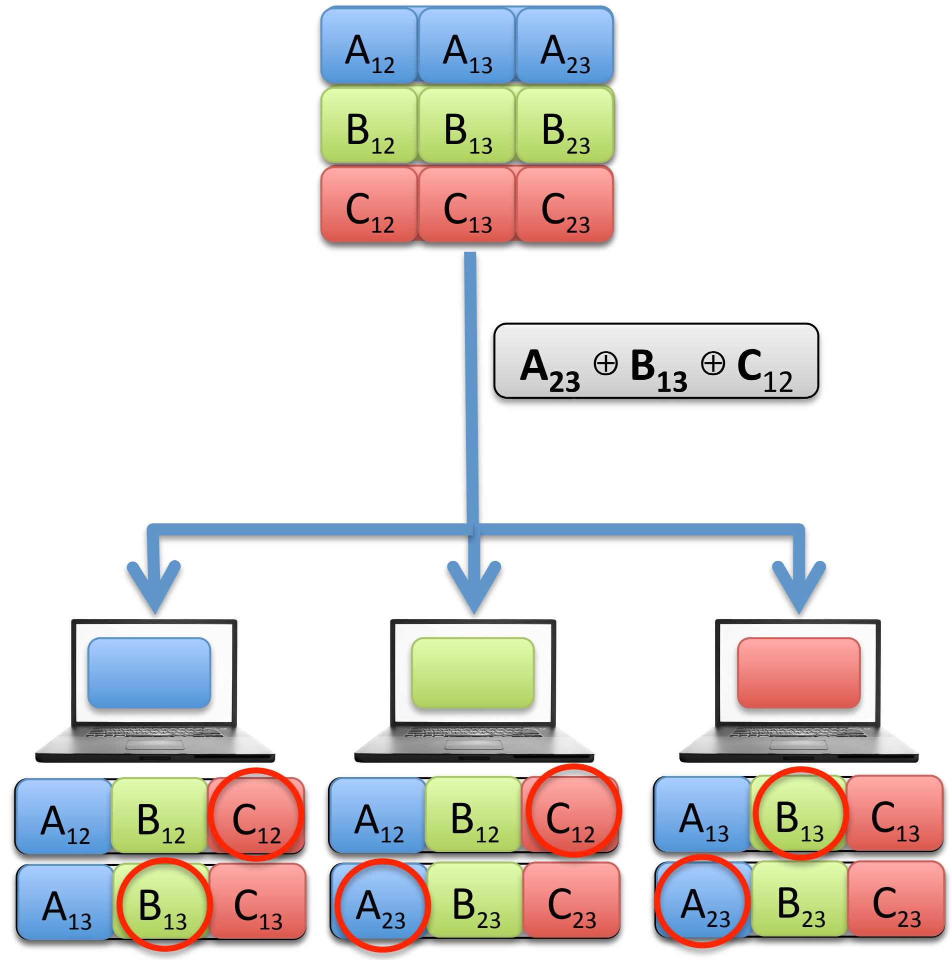


Multicasting Opportunity between two users with different demands

Proposed Scheme

N=3 Files, K=3 Users, Cache Size M=2





Multicasting Opportunity between two users with different demands

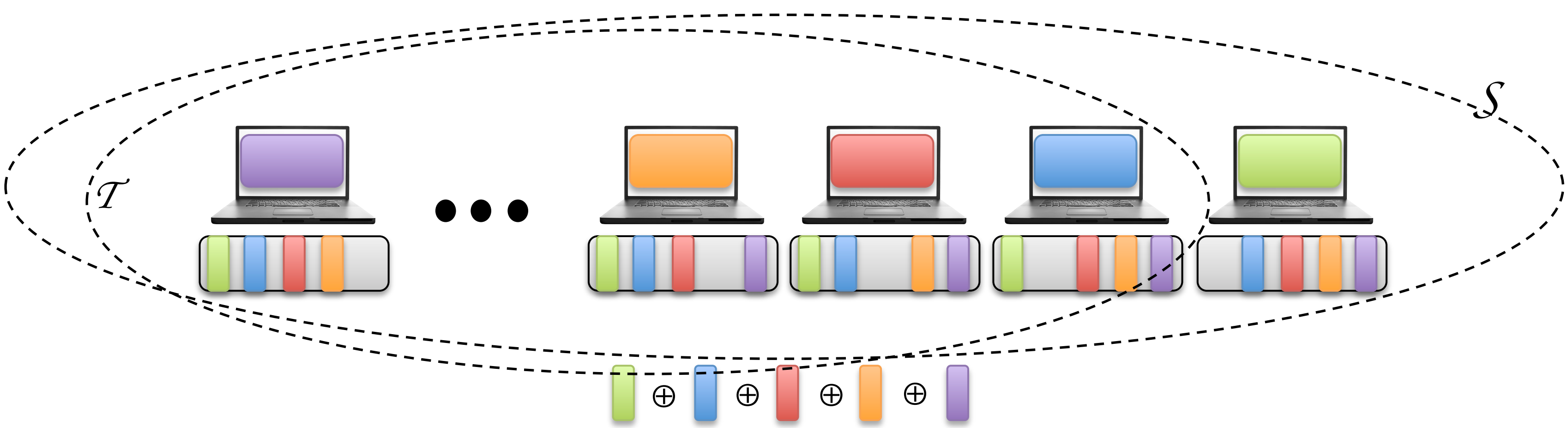
Proposed Scheme

K=N Files and Users, Cache Size M

Objective: Multicast to M+1 users with different demands

Need to place the content such that:

- for every possible set of demands,
- and for every subset $\mathcal{S}$ of $M+1$ users,
- and for every subset $\mathcal{T}$ of $\mathcal{S}$ with M users,
- users in $\mathcal{T}$ share a content required by user $\mathcal{S}\backslash\mathcal{T}$



Proposed Scheme

N=K Files and Users, Cache Size M

-*N* files: $W_1, W_2, \ldots, W_N$

- Split each file into $\binom{K}{M}$ parts

$$\Rightarrow \quad W_n = (W_{n,\mathcal{T}} : \mathcal{T} \subset [K], |\mathcal{T}| = M)$$

- Cache *k:* $(W_{n,\mathcal{T}} : \; n \in [N], \mathcal{T} \subset [K], |\mathcal{T}| = M, k \in \mathcal{T})$

Example: *K=N=3, M=2*

Cache 1=(A_{12}, A_{13}, B_{12}, B_{13}, C_{12}, C_{13})

Proposed Scheme

N=K Files and Users, Cache Size M

- Assume user *k* asks for W_{d_k}

- Send $\oplus_{k\in\mathcal{S}} W_{d_k, S\setminus\{k\}}$ for all $S \subset [K]$ such that $|S| = M+1$

Example: *K=N=3, M=1*

For demands of (A,B,C)

{1,2} ➔ ($A_2 \oplus B_1$)

{1,3} ➔ ($A_3 \oplus C_1$)

{2,3} ➔ ($B_3 \oplus C_2$)

Comparison

N Files, K Users, Cache Size M

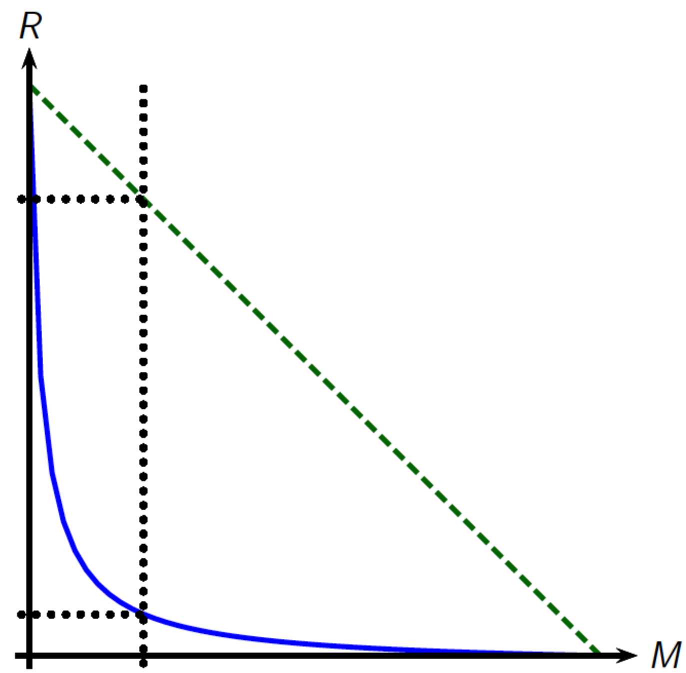
- Conventional scheme: $R(M)=K(1-M/N)$
- Proposed scheme: $R(M)=K(1-M/N)\,(1+KM/N)^{-1}$

- Rate without caching: K
- Local caching gain: $1-M/N$
  - Significant when local cache size M is in the order of N
- Global caching gain: $(1+KM/N)^{-1}$
  - Significant when global cache size KM is in the order of N

Reduction in rate is in the order of number of users.

Comparison

N=50 Files, K=50 Users, Cache Size M=10



- Conventional Scheme:

  $$R(M) = K(1-M/N)$$

  $$= 50 \times 0.8 = 40$$

- Proposed scheme:

  $$R(M) = K(1-M/N)\,(1+KM/N)^{-1}$$

  $$= 50 \times 0.8 \times 0.09 = 3.6$$

- Factor of 11 times improvement
- In the proposed scheme, there is multicasting among 11 users

Can We Do Better?

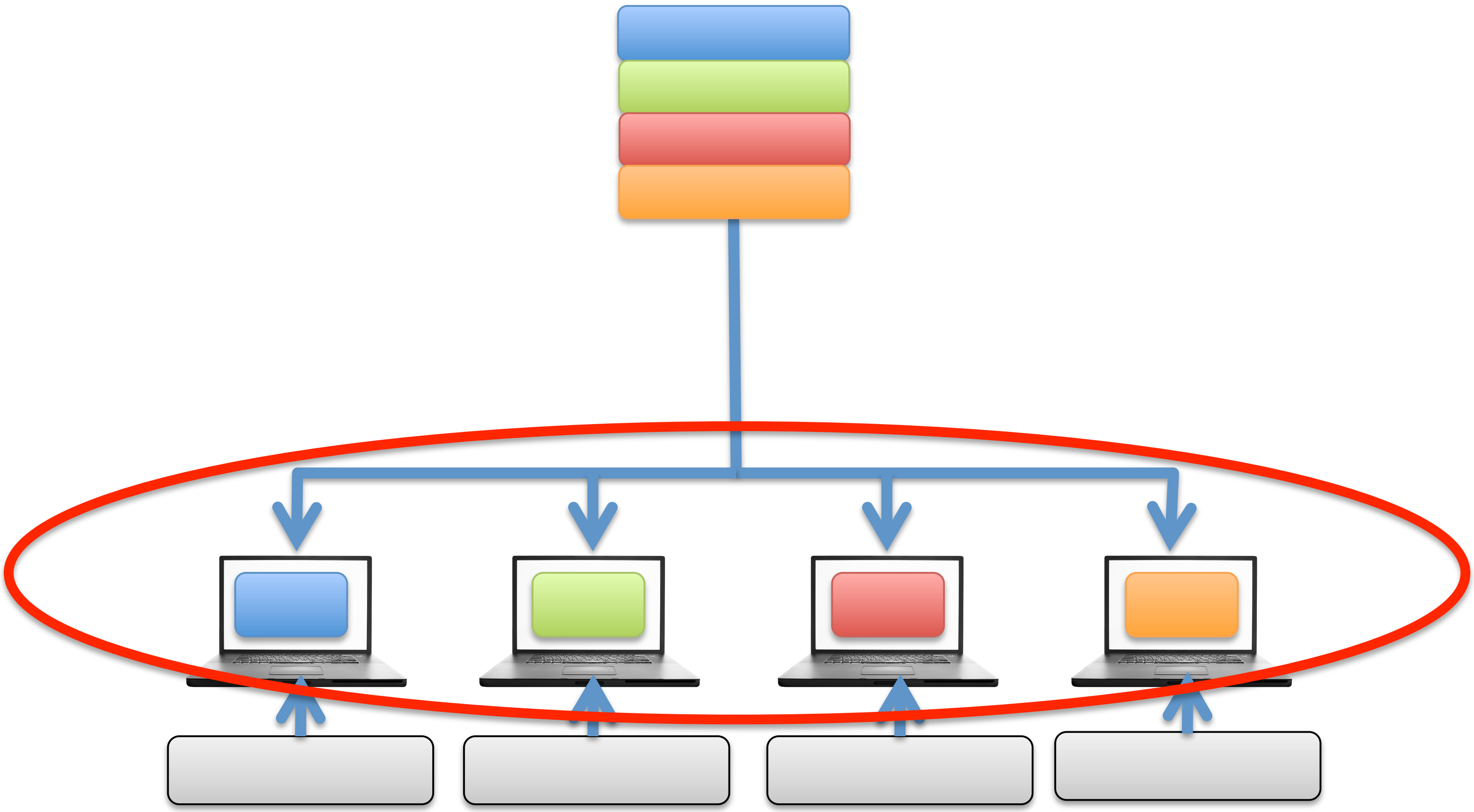
Theorem:

The proposed scheme is optimum within a constant factor in rate.

- Information Theoretic Bound.
- The constant gap is independent of the parameters of the problem.
- No significant gain beside local and global gains.

Outer-Bound

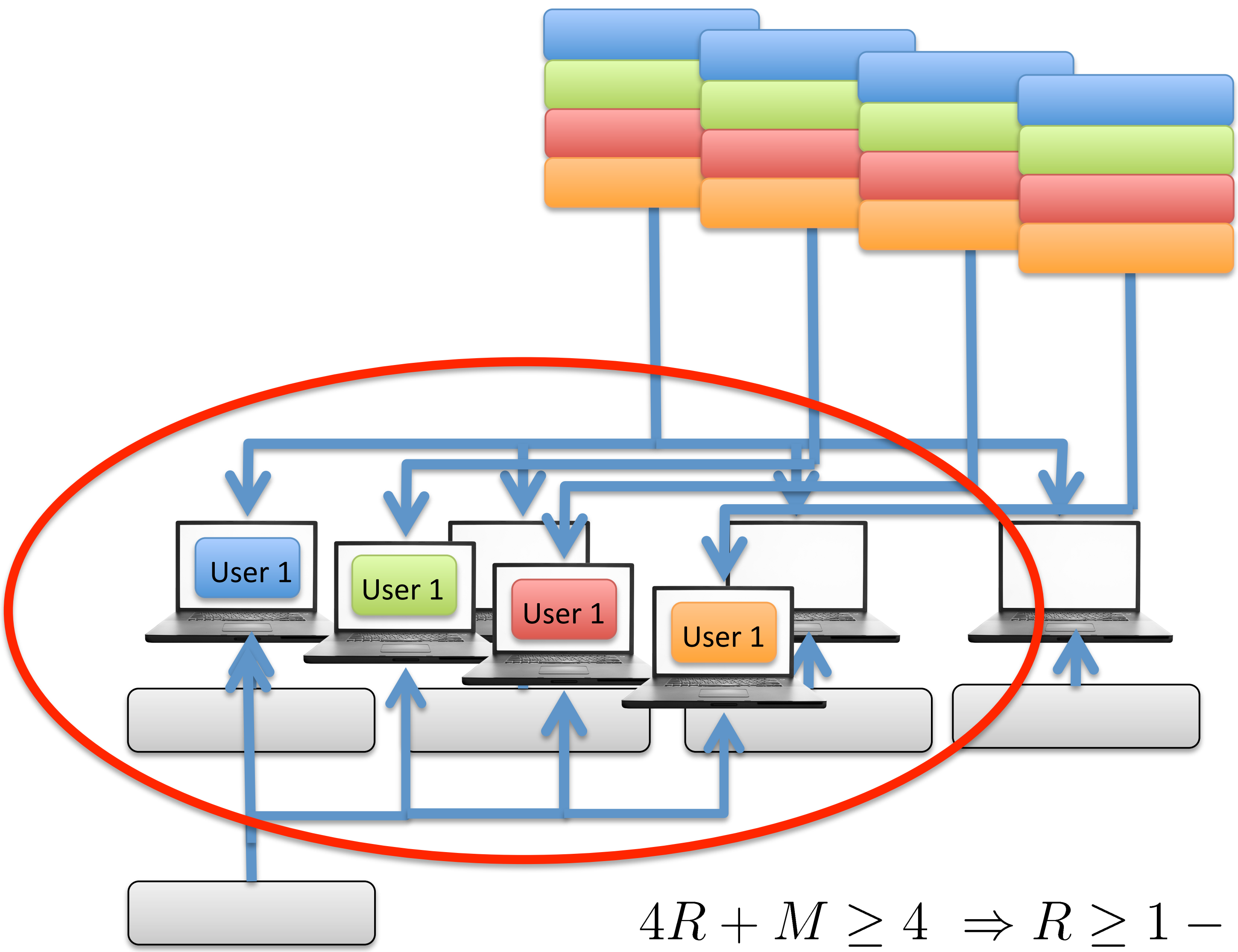
N=4 Files, K=4 Users, Cache Size M



$$R + 4M \geq 4 \;\; \Rightarrow \;\; R \geq 4 - 4M$$

Outer-Bound

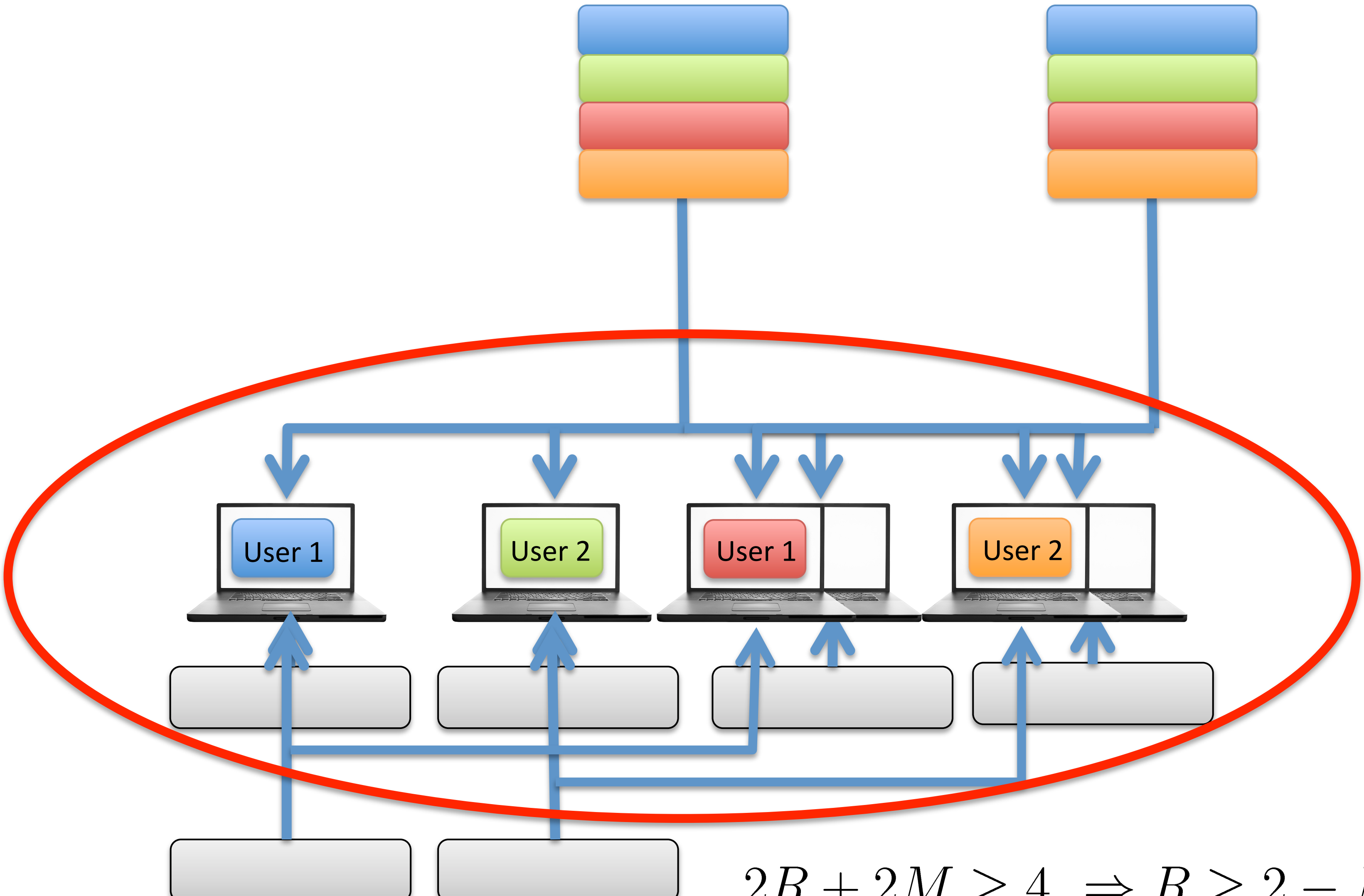
N=4 Files, K=4 Users, Cache Size M



$$4R + M \geq 4 \;\Rightarrow R \geq 1 - M/4$$

Outer-Bound

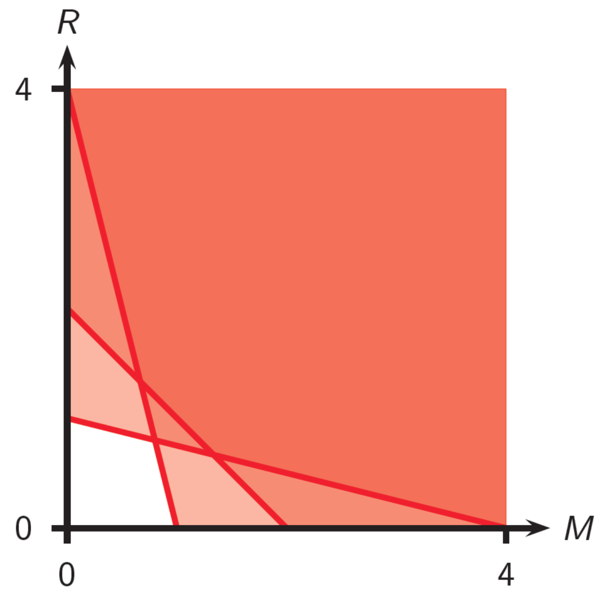
N=4 Files, K=4 Users, Cache Size M



$$2R + 2M \geq 4 \;\; \Rightarrow R \geq 2 - M$$

Outer-Bound

$$R \geq \max\{4 - 4M, 1 - M/4, 2 - M\}$$



For general *K* and *N*,

$$R \geq \max_s \left(s - \frac{s}{\lfloor N/s \rfloor} M\right)$$

Further Questions

- Do we need to coordinate in the placement phase? No
- Do users' request need to be synchronized? No
- Is caching random linear combinations efficient? No

Conclusion

- In early feedback (demands known before prefetching),
  - the main gain of caching is local.
- In late feedback (demands are known after prefetching):
  - The main gain in caching is global.
  - Enabled by Simultaneous multicasting gain among users with different demands, no matter what the demands are.
  - Global cache size matters, even though memories are isolated.
- Papers available on arxiv:
  - Maddah-Ali, Niesen, *Fundamental Limits of Caching*
  - Maddah-Ali, Niesen: *Decentralized caching attains order-optimal memory-rate trade-off*