ITW-2018 TUTORIAL

CODED CACHING AND DISTRIBUTED COMPUTING: OPPORTUNITIES AND CHALLENGES

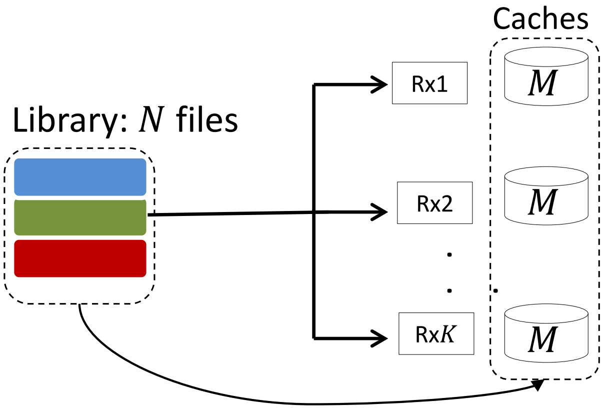
ADITYA RAMAMOORTHY (IOWA STATE UNIVERSITY)

PETROS ELIA (EURECOM – FRANCE)

Two Problems Sharing Some Fundamentals

Cache-aided Communications

Communications in Distributed Computing



Common ingredients:
Communication cost and redundancy of stored/computed data

Common tool: Clique-based coding
Common Bottlenecks/Challenges

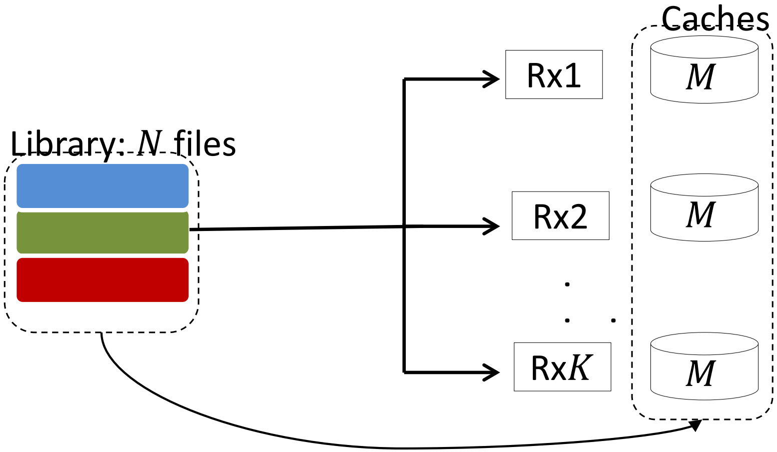
Outline

- Basic elements of coded caching (first communications)
  - ➢ Basic properties
  - ➢ Main gains & Important variants
- Ramifications of coded caching in distributed computing
  - – Coded map reduce
  - – Tradeoff between computing and communicating
  - – Main bottlenecks
- Main challenges
  - – Subpacketization: The bottleneck and new algorithmic solutions
  - – Non uniformity: The bottleneck and new solutions
- Exploiting multiple dimensions to alleviate bottlenecks
  - – Multiplicative gains by reducing subpacketization
  - – Reducing effect of non uniformity
  - – New coding structures
- Open problems and closing remarks

Intro

Simple Caching

First: Cache-Aided Communications

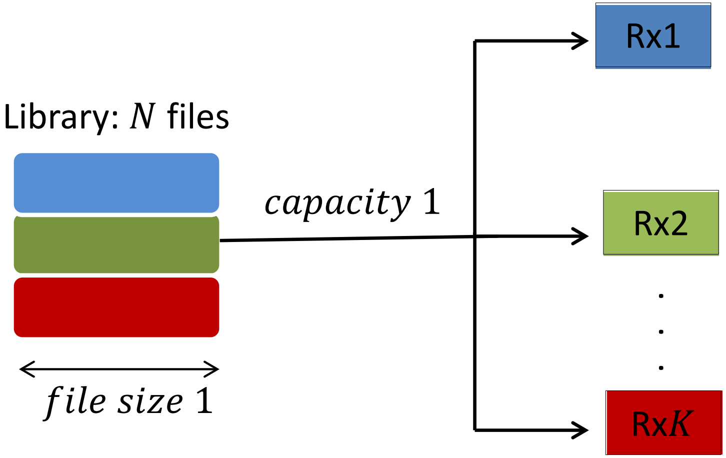


$$\gamma \overset{\text{def}}{=} \frac{M}{N} \overset{\text{def}}{=} \frac{\textit{indivial cache size}}{\textit{library size}}$$

$T(\gamma)$: *duration of delivery phase*

OBJECTIVE: reduce $T(\gamma)$

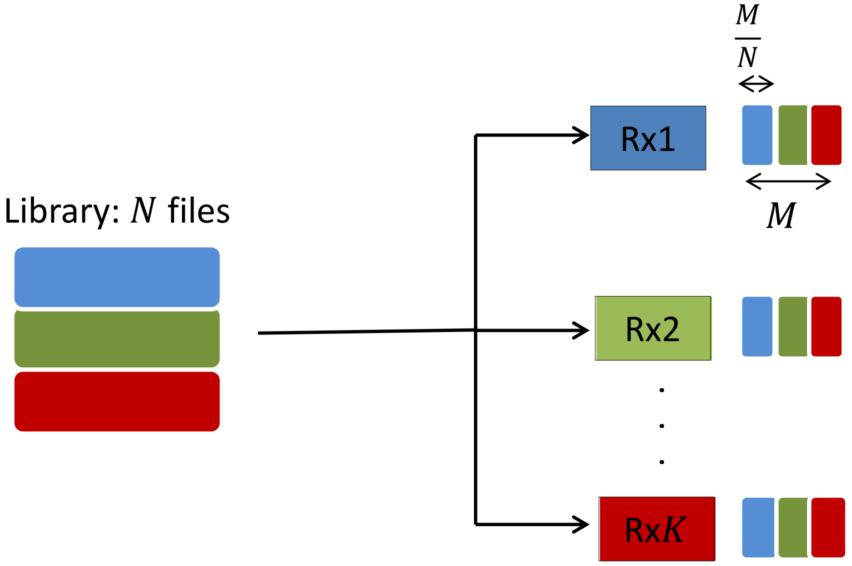
Single-stream Channel: No Caching ($M = 0$)



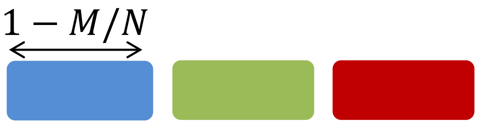
- Transmission sequence: 🟦 🟩 🟥

$$\boxed{T = K}$$

Traditional Caching (Worst-Case Consideration)



- Transmission sequence:

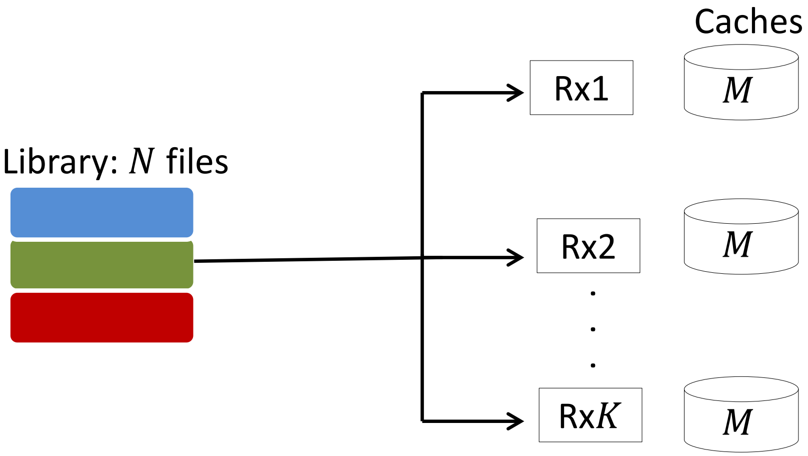


- Local cache gain: $(1-M/N)$ for each user
- The rate:

$$\boxed{T = K(1-M/N) = K(1-\gamma), \qquad \gamma \overset{\text{def}}{=} \frac{M}{N}}$$

Coded Caching

The power of advanced caching

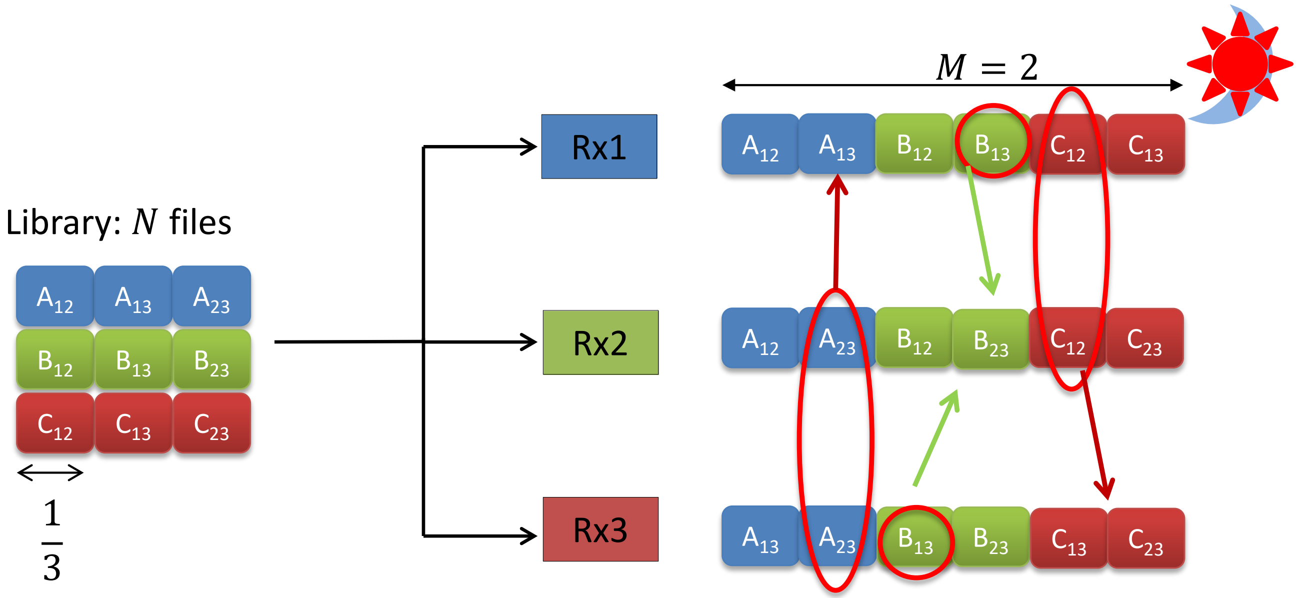


Key breakthrough: USE CACHES TO CANCEL INTERFERENCE

- Cache so that one transmission is useful to many
    - Even if requested files are different
- Large decreases in delay

Result: Maddah-Ali, Niesen (2013)

Example: $N = K = 3, M = 2 \quad (\gamma = \frac{M}{N} = \frac{2}{3})$


Library: N files
A_{12} A_{13} A_{23}
B_{12} B_{13} B_{23}
C_{12} C_{13} C_{23}
$\frac{1}{3}$
Rx1
Rx2
Rx3
$M = 2$
A_{12} A_{13} B_{12} B_{13} C_{12} C_{13}
A_{12} A_{23} B_{12} B_{23} C_{12} C_{23}
A_{13} A_{23} B_{13} B_{23} C_{13} C_{23}

- Transmit : $A_{23} \oplus B_{13} \oplus C_{12}$ (a common message for all)

$$\boxed{Gain = 3 = \frac{KM}{N} \; users\ at\ a\ time}$$

Coded Caching: Intuition - Clique

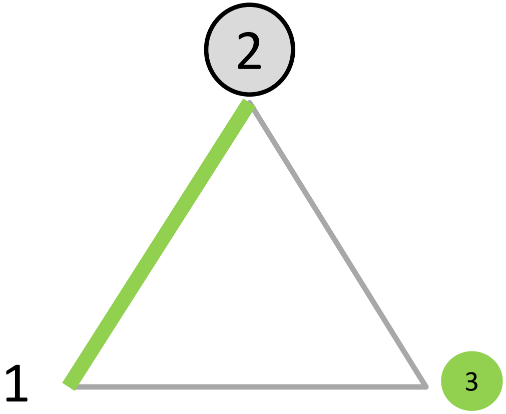
- *Deliver to $K\gamma + 1$ users at a time*

- *Via XORs with $K\gamma + 1$ subfiles.*
    - ***Each user (out of the $K\gamma + 1$ now served) knows all summands except its own***

$$T = \frac{K(1-\gamma)}{1+K\gamma} \approx \frac{1-\gamma}{\gamma}$$

Coded Caching: Intuition - Clique

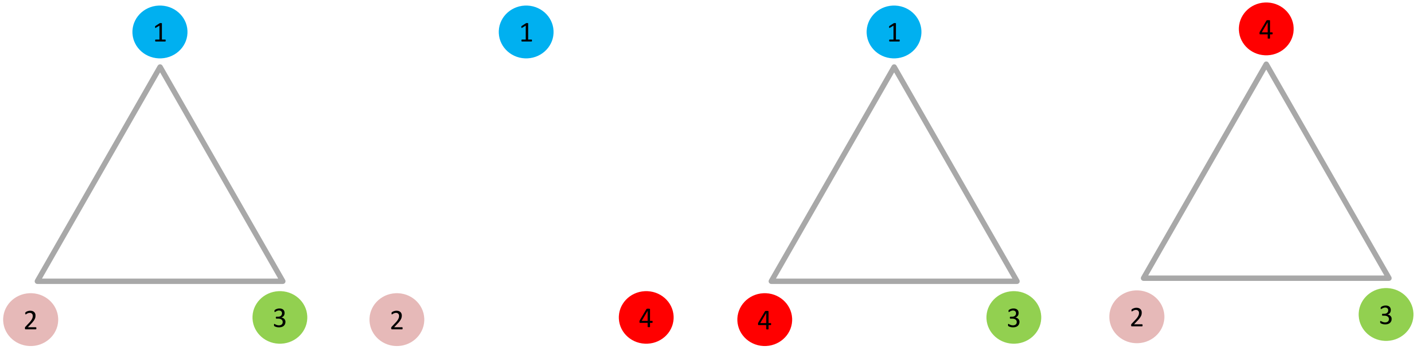
$$K = 3, K\gamma = 2$$



$$A_{23} \oplus B_{13} \oplus C_{12}$$

Coded Caching: Intuition - Cliques

$$K = 4, K\gamma = 2$$



$A_{23} \oplus B_{13} \oplus C_{12}$

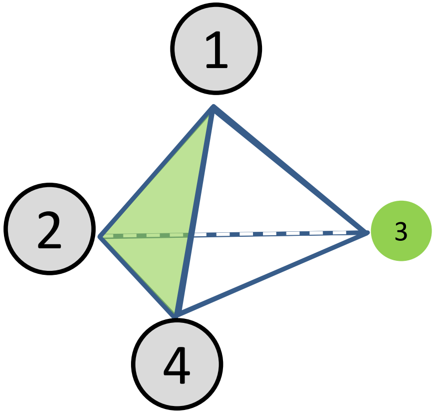
$A_{24} \oplus B_{14} \oplus D_{12}$

$A_{34} \oplus D_{13} \oplus C_{14}$

$D_{23} \oplus B_{34} \oplus C_{24}$

Coded Caching: Intuition - Clique

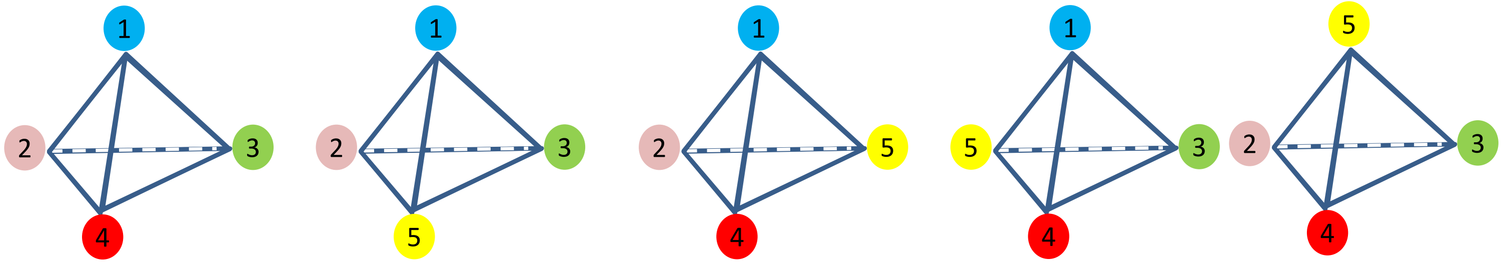
$$K = 4, K\gamma = 3$$



$$D_{123} \oplus A_{234} \oplus B_{134} \oplus C_{124}$$

Coded Caching: Intuition - Cliques

$$K = 5, K\gamma = 3$$



$D_{123} \oplus A_{234} \oplus B_{134} \oplus C_{124}$

$E_{123} \oplus A_{235} \oplus B_{135} \oplus C_{125}$

$E_{124} \oplus A_{245} \oplus D_{125} \oplus B_{145}$

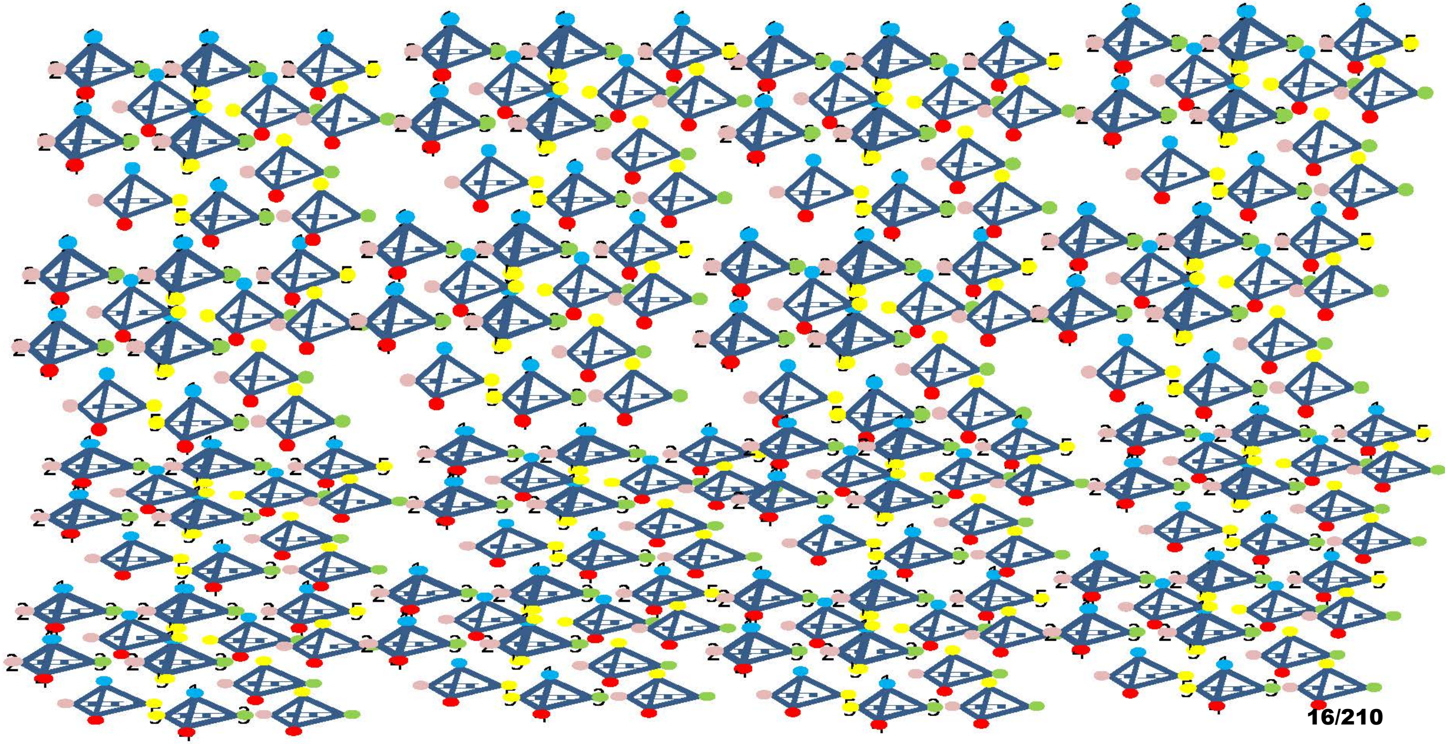
$E_{134} \oplus A_{345} \oplus D_{135} \oplus C_{145}$

$E_{234} \oplus B_{345} \oplus D_{235} \oplus C_{245}$

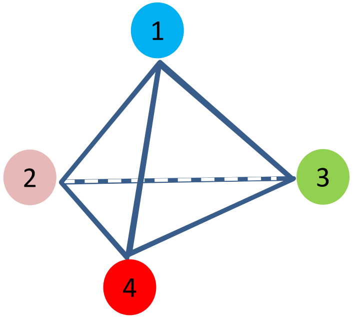
Coded Caching: Intuition – Problematically Many Cliques

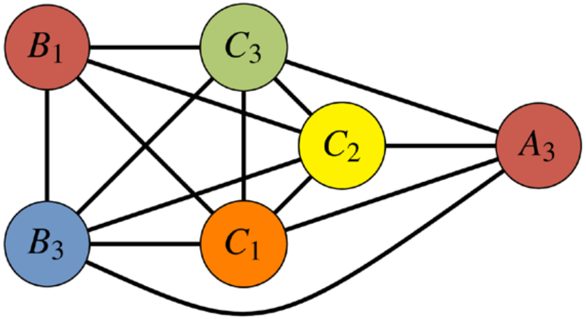
$K = 100, \qquad K\gamma = 9$

$9\ users\ at\ a\ time\ , \qquad 2 \cdot 10^{13}\ \ cliques$



Extension of Clique-based Idea





$D_{123} \oplus A_{234} \oplus B_{134} \oplus C_{124}$

$A_3 \oplus B_1$ B_3 C_1 C_2 C_3

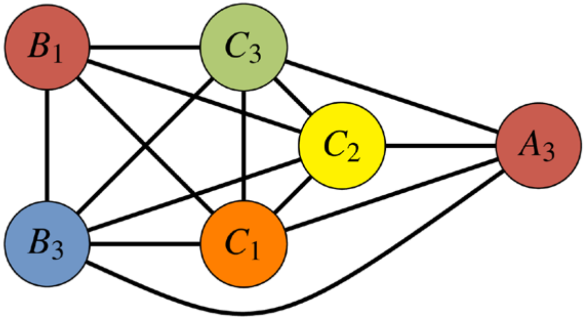
Variety of settings:

- Non-uniform demands
- Decentralized placement

Ji, Tulino, Llorca, Caire (Trans IT 2017)

Non-uniform Example: Graph Coloring for XORs

Connected vertices must have different colors



Transmission

$A_3 \oplus B_1$ B_3 C_1 C_2 C_3

$$\boxed{T(\gamma) = 5/3}$$

Gain

$$\frac{|Q|}{\chi(H_{C,Q})}$$ (χ is chromatic number)

Calculation:

$$\frac{|Q|}{\chi(H_{C,Q})} = \frac{K(1-\gamma)}{T}$$

$$= \frac{3\left(1-\frac{1}{3}\right)}{5/3} = \frac{6}{5}$$

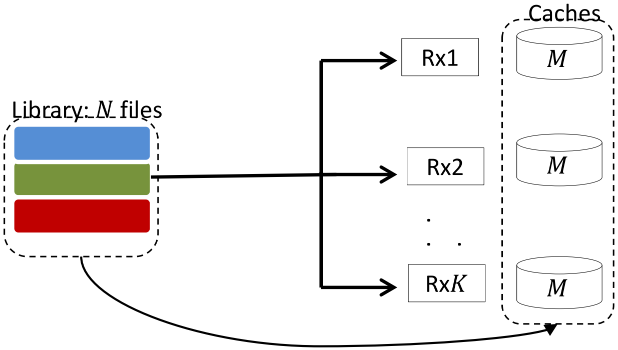
The Optimization Problem

The Optimization Problem:

Optimize how you cache and send each bit

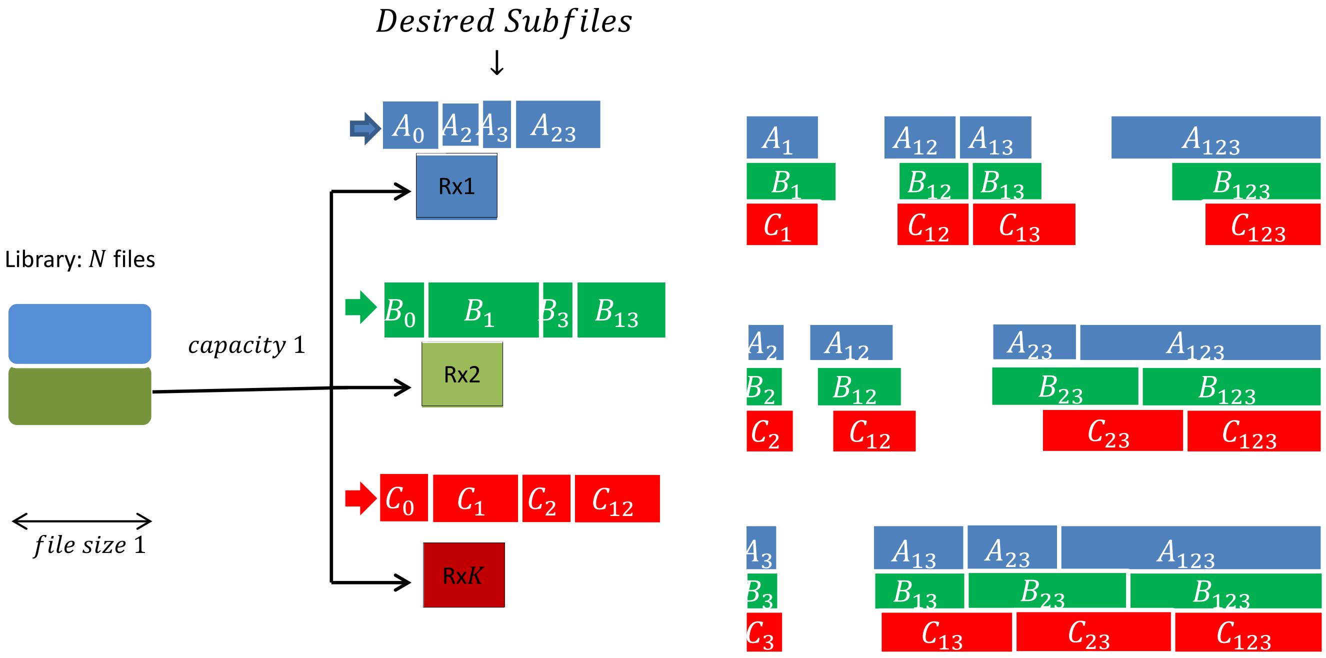
Another (ongoing) Success of Information Theory

The Optimization Problem (Caching Placement Part)



$$T^*(M) = \min_{schemes\ \chi} \max_{demands\ \mathbf{d}} T(\mathbf{d}, \chi, M)$$

The Optimization Problem (Delivery Part)



Desired Subfiles
A_0 A_2 A_3 A_{23}
Rx1
Library: N files
capacity 1
B_0 B_1 B_3 B_{13}
Rx2
C_0 C_1 C_2 C_{12}
RxK
file size 1
A_1 A_{12} A_{13} A_{123}
B_1 B_{12} B_{13} B_{123}
C_1 C_{12} C_{13} C_{123}
A_2 A_{12} A_{23} A_{123}
B_2 B_{12} B_{23} B_{123}
C_2 C_{12} C_{23} C_{123}
A_3 A_{13} A_{23} A_{123}
B_3 B_{13} B_{23} B_{123}
C_3 C_{13} C_{23} C_{123}

Transmission Scheme F → $F_1(A_{P_1}, B_{Q_1}, C_{R_1})$ $F_2(A_{P_2}, B_{Q_2}, C_{R_2})$ ..... $F_{T_F}(A_{P_T}, B_{Q_T}, C_{R_T})$

Scheme Φ → $\Phi_1(A_{P_1}, B_{Q_1}, C_{R_1})$ $\Phi_2(A_{P_2}, B_{Q_2}, C_{R_2})$ ..... $\Phi_{T_\Phi}(A_{P_T}, B_{Q_T}, C_{R_T})$

Scheme Ж → $\text{Ж}_1(A_{P_1}, B_{Q_1}, C_{R_1})$ $\text{Ж}_2(A_{P_2}, B_{Q_2}, C_{R_2})$ ..... $\text{Ж}_{T_\text{Ж}}(A_{P_T}, B_{Q_T}, C_{R_T})$

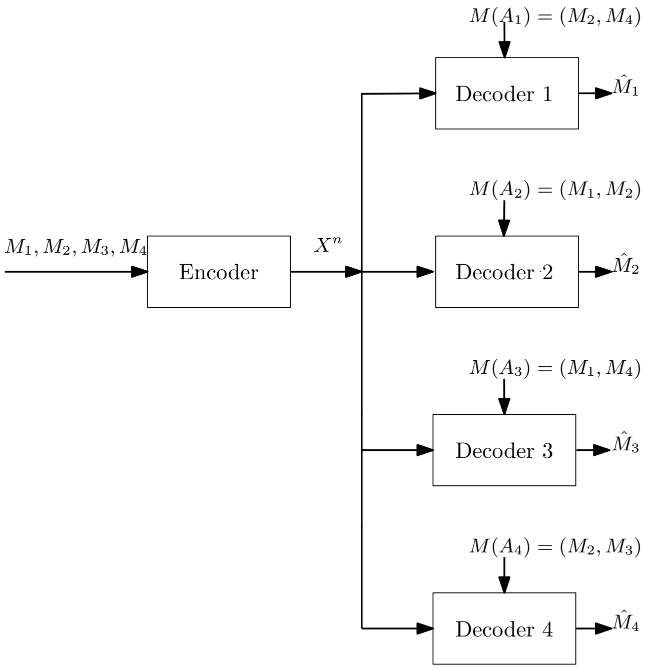
⋮

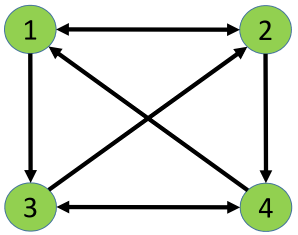
⋮

⋮

Tool: Coded Caching → Index Coding → Graphs

- This is an index coding problem
- Example: 4-message index coding problem





Tool: Maximum Acyclic Subgraph Bound

Theorem

If G is acyclic, then optimal delay is

$$T^*(G) = |G|.$$

Figure: As if no side information (i.e., TDMA is best you can do)

Theorem

For any problem corresponding to an arbitrary G, then

$$T \geq \sum_{j \in J} |M_j|$$

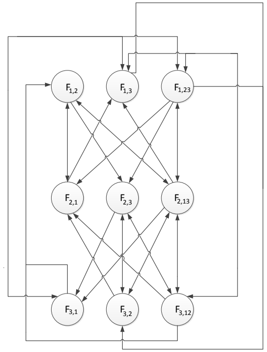
for all acyclic subgraphs $J \subset G$.

1

[1] Arbabjolfaei et al., 2013.

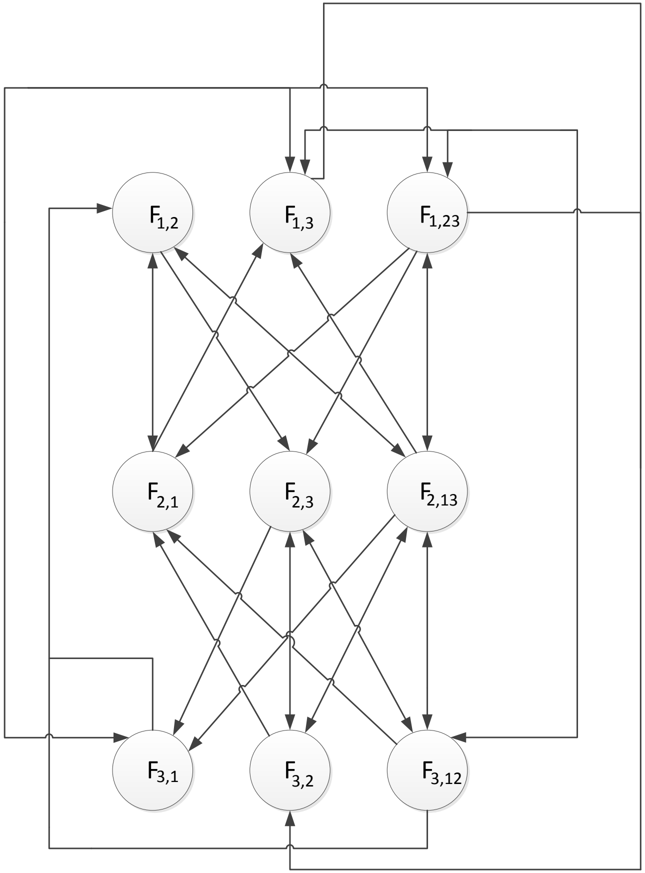
Step: Create Graph for One Caching Problem

- Get $\mathbf{d}$ (e.g. $\mathbf{d} = (1, 2, 3)$)
- General split F_i to $F_{i,\mathcal{W}}$
  - $\mathcal{W} \in 2^{[3]} = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$
- Each node = desired sub-file
- Row j for user j
- Edge from i to j if user j knows sub-file i



- K. Wan, D. Tuninetti and P. Piantanida, "On the Optimality of Uncoded Cache Placement", 2016

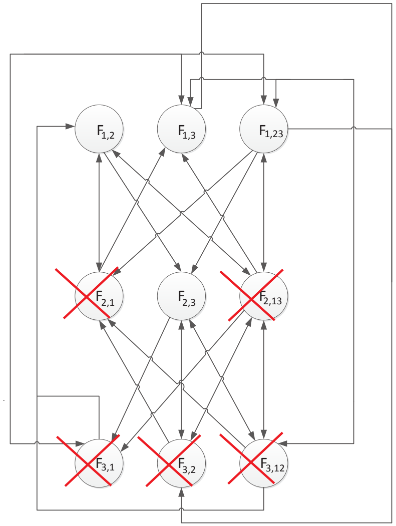
Step: Create Graph for One Caching Problem



[1] Image source: Wan et al. 2017

Step: Create Maximal Acyclic Subgraph - Arrows Down

$$F_{d_1,\mathcal{W}_1} \text{for all } \mathcal{W}_1 \subseteq [1:3] \setminus \{1\}$$
$$F_{d_2,\mathcal{W}_2} \text{for all } \mathcal{W}_2 \subseteq [1:3] \setminus \{1,2\}$$
$$F_{d_3,\mathcal{W}_3} \text{for all } \mathcal{W}_3 \subseteq [1:3] \setminus \{1,2,3\} = \emptyset$$

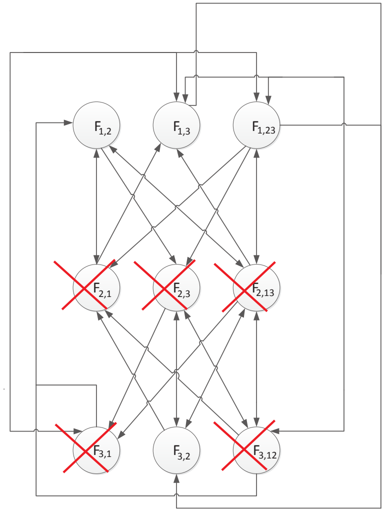


$$T_u(M) \geq |F_{1,\emptyset}| + |F_{1,2}| + |F_{1,3}| + |F_{1,23}| + |F_{2,\emptyset}| + |F_{2,3}| + |F_{3,\emptyset}|$$

Step: More Acyclic Subgraphs - Permute Users $(\pi = (1, 3, 2))$

$$d_{\pi_1} = d_1 = 1; \mathcal{W}_1 \subseteq [1:3] \setminus \{\pi_1\} = \{2, 3\}$$
$$d_{\pi_2} = d_3 = 3; \mathcal{W}_1 \subseteq [1:3] \setminus \{\pi_1, \pi_2\} = \{2\}$$
$$d_{\pi_3} = d_2 = 2; \mathcal{W}_3 \subseteq [1:3] \setminus \{\pi_1, \pi_2, \pi_3\} = \emptyset$$

$$T_u(M) \geq |F_{1,\emptyset}| + |F_{1,2}| + |F_{1,3}| + |F_{1,23}| + |F_{3,2}| + |F_{3,\emptyset}| + |F_{2,\emptyset}|$$



Step: More Graphs for Symmetry - Add Acyclic Subgraphs

- $[3!]$ possible demand vectors
- $[3!]$ possible permutations
- sum $[3!]^2$ possible bounds

$$T_u(M) \geq |F_{1,\emptyset}| + |F_{1,2}| + |F_{1,3}| + |F_{1,23}| + |F_{2,\emptyset}| + |F_{2,3}| + |F_{3,\emptyset}|$$

$$T_u(M) \geq |F_{1,\emptyset}| + |F_{1,2}| + |F_{1,3}| + |F_{1,23}| + |F_{3,2}| + |F_{3,\emptyset}| + |F_{2,\emptyset}|$$

$$\vdots \;\; \vdots$$

$$T_u(M)(3!)^2 \geq \sum_{\mathbf{d}} \sum_{\pi} \sum_{j \in [3]} \sum_{\mathcal{W}_j \in 2^{[3]} : \mathcal{W}_j \setminus \{\pi_1, \cdots, \pi_j\}} |F_{d_{\pi_j}, \mathcal{W}_j}|$$

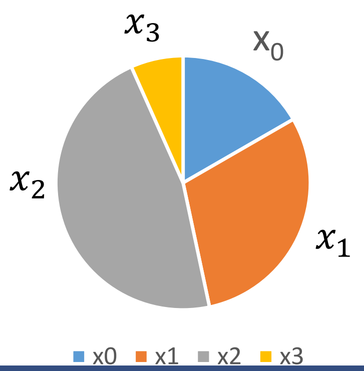
Step: Exploit Symmetry - Counting Arguments

- Each sub-file that is cached in $|\mathcal{W}| = t$ caches, appears an equal number of times
- Allows for nice transition, from

$$T_u(M) \geq \sum_{j \in [3]} \sum_{\mathcal{W}_j \in 2^{[3]} : \mathcal{W}_j \setminus \{\pi_1, \cdots, \pi_j\}} |F_{d_{\pi_j}, \mathcal{W}_j}|$$

- to

$$T_u(M) \geq \sum_{i \in [0:3]} x_i \frac{1 - i/3}{1 + i} = x_0 + \frac{1}{3} x_1 + \frac{1}{9} x_2 + 0 x_3 \qquad (1)$$



Step: Final optimization

- Optimize

$$x_0 + \frac{1}{3} \cdot x_1 + \frac{1}{9} \cdot x_2 + 0 \cdot x_3$$

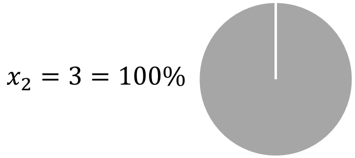
- Under total file size constraint:

$$x_0 + x_1 + x_2 + x_3 = 3$$

- total cache size constraint:

$$0 \cdot x_0 + 1 \cdot x_1 + 2x_2 + 3 \cdot x_3 \leq 3M$$

- Solution (*See also Yu, Maddah-Ali, Avestimehr*):



- Final answer:

$$T^* \geq \frac{K-t}{1+t} = \frac{K(1-\gamma)}{1+K\gamma}$$

Toward average delay: N<K

- Optimality result extends to case of $N < K$

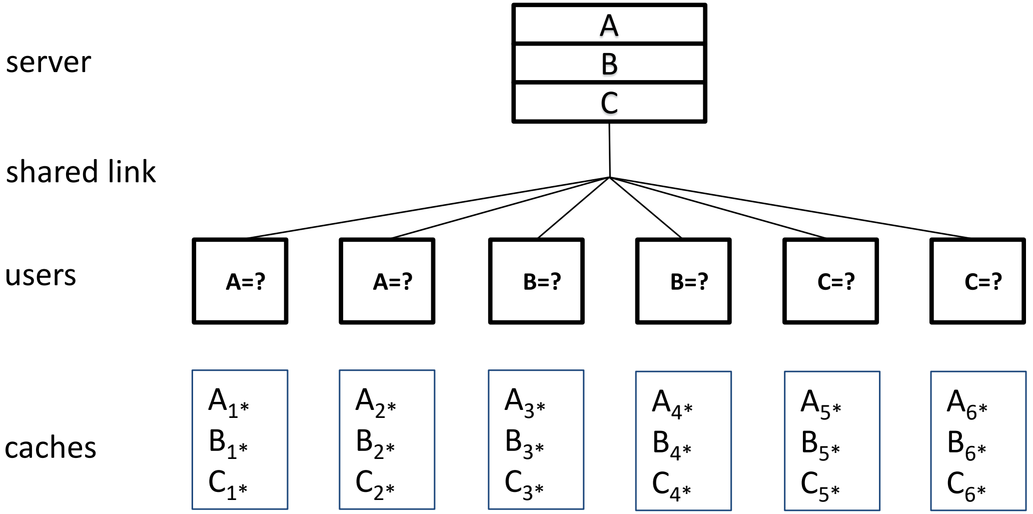
$$T^* = \frac{\binom{K}{K\gamma+1} - \binom{K-\min(K,N)}{K\gamma+1}}{\binom{K}{K\gamma}}$$

- As well as to average-delay case for uniform distribution
  - Single stream scenario

- Proof based on new scheme for $N < K$

Yu, Maddah-Ali, Avestimehr TiT 2018

Toward average delay: N<K


server
A
B
C
shared link
users
A=?
A=?
B=?
B=?
C=?
C=?
caches
A_{1*} B_{1*} C_{1*}
A_{2*} B_{2*} C_{2*}
A_{3*} B_{3*} C_{3*}
A_{4*} B_{4*} C_{4*}
A_{5*} B_{5*} C_{5*}
A_{6*} B_{6*} C_{6*}

- $K = 6, N = 3, M = 1$
- Subpacketization $\binom{6}{2}$ = 15

  A_{12}, A_{13}, A_{14}, …, A_{56},
  B_{12}, …, B_{56}, C_{12}, …, C_{56}.

- MN Placement

- MN Delivery:

Broadcast $\binom{6}{3}$ = 20 XORs $\Rightarrow T = \frac{20}{15}$.

- Modified scheme can skip 1 XOR

$$\Rightarrow T = \frac{19}{15}$$ Optimal

Image-source and Example: Yu et al.

Toward average delay: N<K

- Ordinarily would send 20 XORs

- $X_{123} = A_{23} \oplus A_{13} \oplus \mathbf{B_{12}}$
- $X_{124} = A_{24} \oplus A_{14} \oplus \mathbf{B_{12}}$
- $X_{125} = A_{25} \oplus A_{15} \oplus \mathbf{C_{12}}$
- $X_{126} = A_{46} \oplus A_{26} \oplus \mathbf{C_{12}}$
- .....
- $X_{146} = \mathbf{A_{46}} \oplus B_{16} \oplus C_{14}$
- .....
- ~~$X_{246} = \mathbf{A_{46}} \oplus B_{26} \oplus$~~ C_{24}
- .....
- $X_{456} = B_{56} \oplus C_{46} \oplus C_{45}$

Example: Yu et al.

Toward average delay: N<K

- Skip $\mathbf{X_{246}} = A_{46} \oplus B_{26} \oplus C_{24}$
- Focus on user 2 (needs $\mathbf{A_{46}}$)

- Have transmitted:
  - $X_{146} = A_{46} \oplus B_{16} \oplus C_{14}$
  - $X_{145} = A_{45} \oplus B_{15} \oplus C_{14}$
  - $X_{136} = A_{36} \oplus B_{16} \oplus C_{13}$
  - $X_{135} = A_{35} \oplus B_{15} \oplus C_{13}$

- Adding up:
  - $X_{146} \oplus X_{145} \oplus X_{136} \oplus X_{135} = A_{46} \oplus \mathbf{A_{45}} \oplus \mathbf{A_{36}} \oplus \mathbf{A_{35}}$

- $A_{45} \leftarrow X_{245} = A_{45} \oplus B_{25} \oplus C_{24}$
- $A_{36} \leftarrow X_{236} = A_{36} \oplus B_{26} \oplus C_{23}$
- $A_{35} \leftarrow X_{235} = A_{35} \oplus B_{25} \oplus C_{23}$

- ➔ Decode $A_{\{4,6\}}$ (➔ similarly B_{26} for user 4, C_{24} for user 6).

Example: Yu et al.

Extensions wide open

- This is for the very special case of the single stream Broadcast Channel pause
- Approach requires new tools when setting changes slightly
  - Shared caches (Hachem, Karamchandrani et Diggavi)
  - multiple file demands (Wei-Ulukus)
  - Multi-layer files (high-def, low-def) (Yang-Gündüz)
- Need methods needed that preserve topology, and non-uniformity
- Need methods for reflecting multiple-transmitters/multiple-antennas.

Coded Distributed Computing: Another side of the same coin ...

s

BigData Analytics and Distributed Computing

Distributed Computing Platforms

BigData Analytics

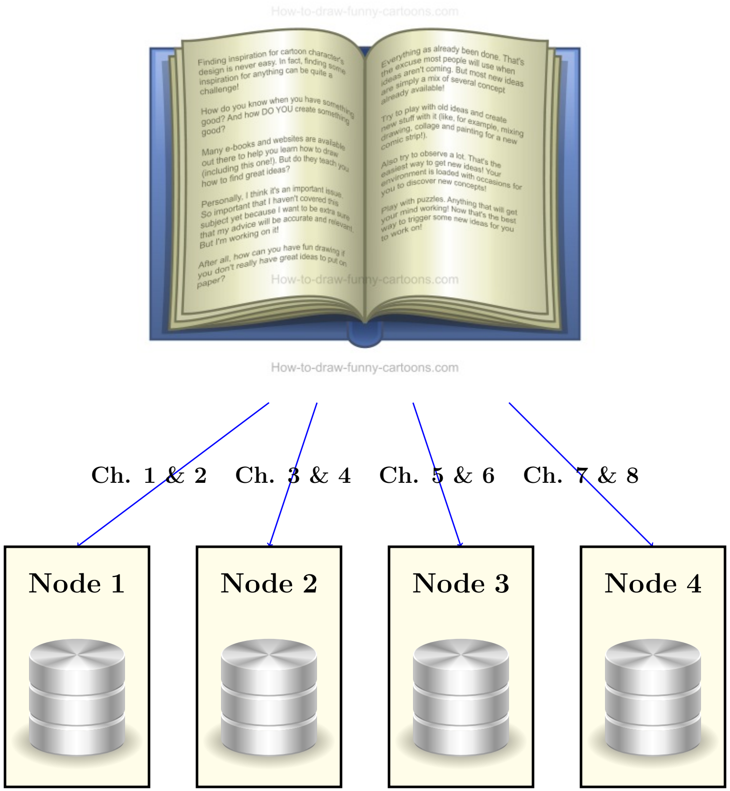
Distributed Computing Platforms

- BigData Analytics often have data that too large to be stored on one machine.
  - **Distributed computing is necessity, not a luxury.**
- Hadoop MapReduce, Apache Spark etc. are popular platforms that allow the end user to use a computing cluster for running large scale jobs.

Steps in the Map-Reduce pipeline

- **MAP:** Split the job/data into smaller pieces. Each part goes to a node (mapper) who work in parallel.
- **SHUFFLE:** Upon completion of the MAP step, nodes exchange appropriate information between themselves so that they can complete the execution.
- **REDUCE:** Each node works (in parallel) on its stored data and data received in the shuffle phase to complete the overall job.

Classic Example: Word Counting in a book



How-to-draw-funny-cartoons.com

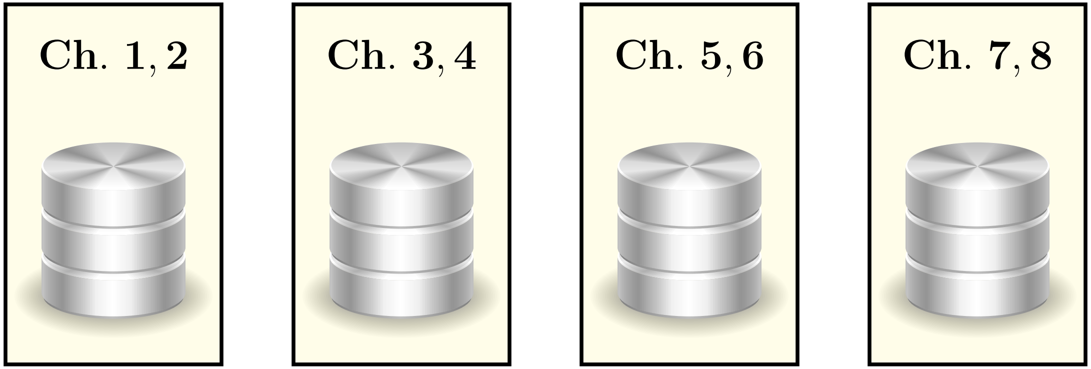
Ch. 1 & 2 Ch. 3 & 4 Ch. 5 & 6 Ch. 7 & 8

Node 1 Node 2 Node 3 Node 4

Classic Example: Word Counting in a book

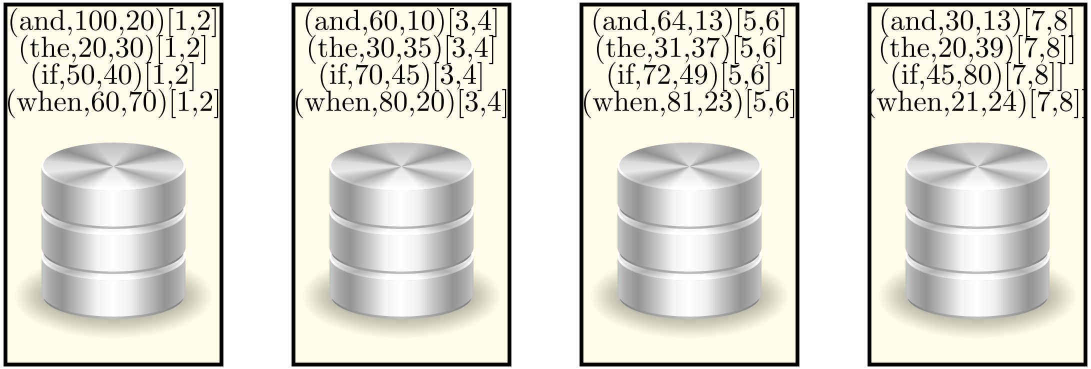
- Suppose that we want to count the occurrences of the words: “and",“the",“if" and “when" in a book with 8 chapters.

- We use four servers, each server is assigned two chapters.



Classic Example: Word Counting in a book

MAP PHASE



(and,100,20)[1,2]
(the,20,30)[1,2]
(if,50,40)[1,2]
(when,60,70)[1,2]

(and,60,10)[3,4]
(the,30,35)[3,4]
(if,70,45)[3,4]
(when,80,20)[3,4]

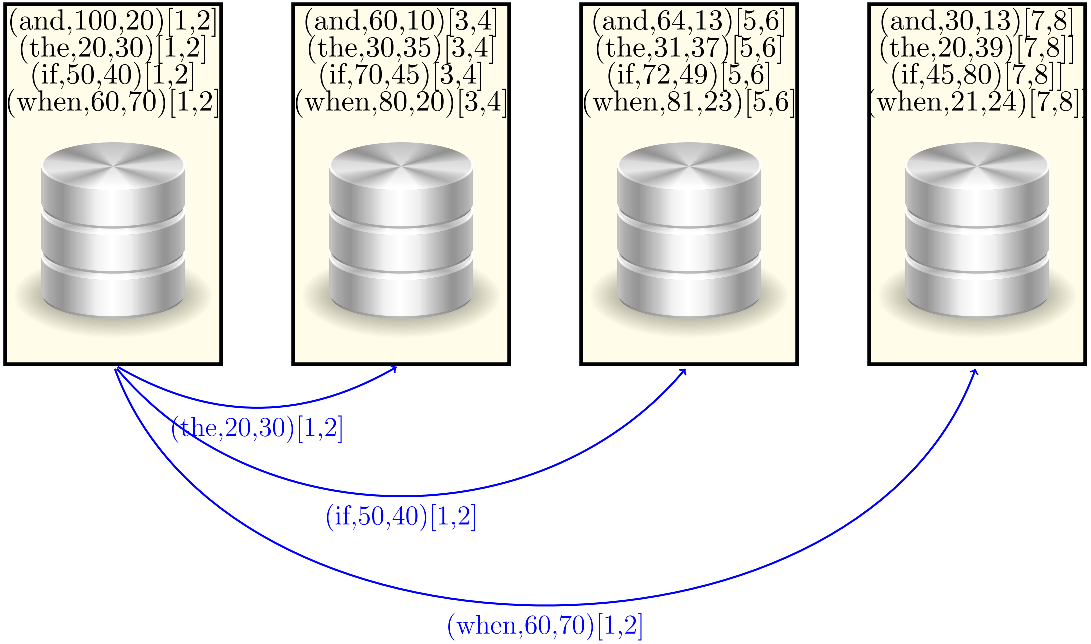
(and,64,13)[5,6]
(the,31,37)[5,6]
(if,72,49)[5,6]
(when,81,23)[5,6]

(and,30,13)[7,8]
(the,20,39)[7,8]]
(if,45,80)[7,8]]
(when,21,24)[7,8]]

Classic Example: Word Counting in a book

SHUFFLE PHASE

3 transmissions from node 1



(and,100,20)[1,2]
(the,20,30)[1,2]
(if,50,40)[1,2]
(when,60,70)[1,2]

(and,60,10)[3,4]
(the,30,35)[3,4]
(if,70,45)[3,4]
(when,80,20)[3,4]

(and,64,13)[5,6]
(the,31,37)[5,6]
(if,72,49)[5,6]
(when,81,23)[5,6]

(and,30,13)[7,8]
(the,20,39)[7,8]]
(if,45,80)[7,8]]
(when,21,24)[7,8]

(the,20,30)[1,2]

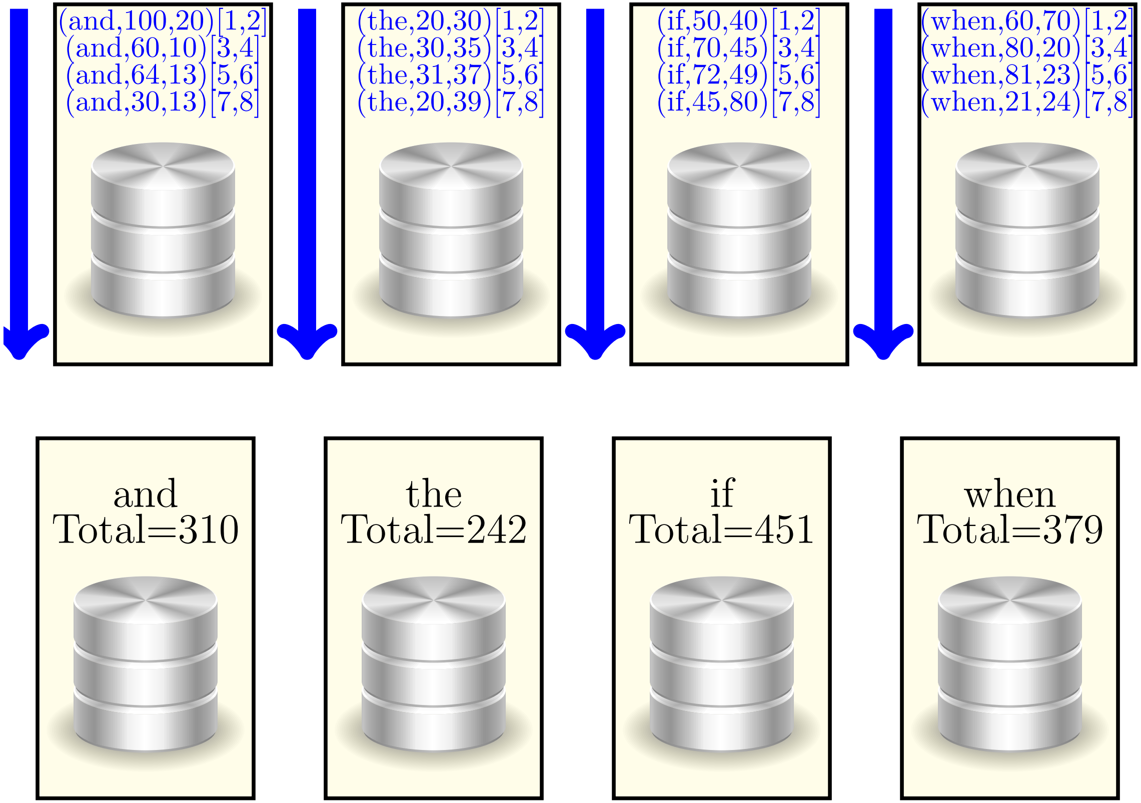
(if,50,40)[1,2]

(when,60,70)[1,2]

3 transmissions from node 2

Classic Example: Word Counting in a book

REDUCE PHASE



(and,100,20)[1,2]
(and,60,10)[3,4]
(and,64,13)[5,6]
(and,30,13)[7,8]

(the,20,30)[1,2]
(the,30,35)[3,4]
(the,31,37)[5,6]
(the,20,39)[7,8]

(if,50,40)[1,2]
(if,70,45)[3,4]
(if,72,49)[5,6]
(if,45,80)[7,8]

(when,60,70)[1,2]
(when,80,20)[3,4]
(when,81,23)[5,6]
(when,21,24)[7,8]

and
Total=310

the
Total=242

if
Total=451

when
Total=379

END OF COMPUTATION

Classic Example: Observations

- Typically, Map and Reduce phases are “Computation Intensive".
- In contrast, Shuffle phase is “Communication Intensive".
    - In Word Count, each server sends three messages to others.
    - Depending upon application, these network transfers can be rather time-consuming.

Terasort & Machine Learning

- A large number of records that are indexed by (key, value) pairs.
- Need to be sorted by key values in a distributed fashion.

Communication phase poses a significant bottleneck for the algorithm.

> "... *these transfers can have a significant impact on job performance* accounting for more than 50% of the job completion times ..." [1]

188,000 MapReduce jobs on a Facebook Hadoop cluster in a week

33% of running time spent on shuffling (on average).

[1] M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica, Managing Data Transfers in Computer Clusters with Orchestra, *SIGCOMM Comput. Commun. Rev.*, Aug. 2011.

Experimental Results on Amazon Web Services (AWS)

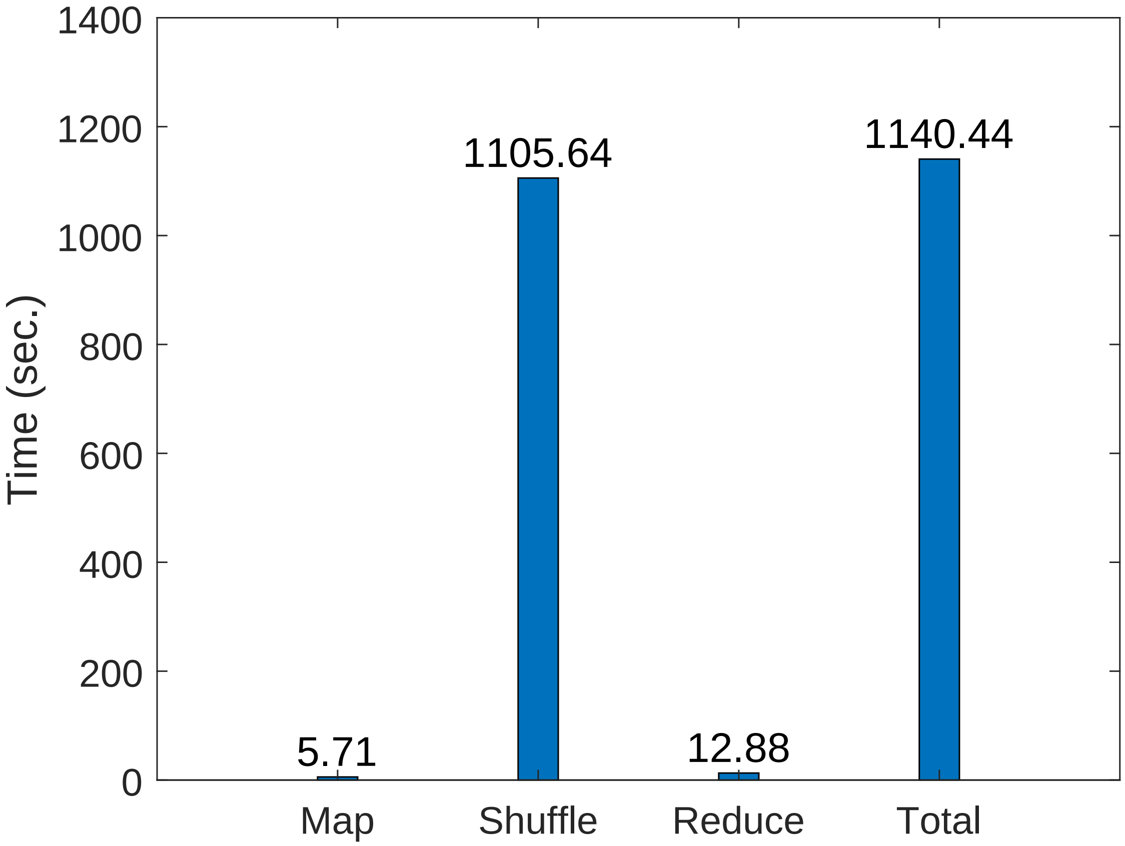
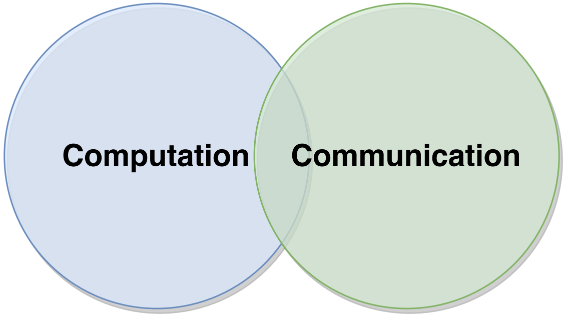


Figure: Uncoded TeraSort on a 12GB dataset with 16 worker nodes.

Computation-Communication tradeoff



- Natural tradeoff between communication and computation (*inversely proportional*)[2].

How can we optimally trade off computing load and communication load?

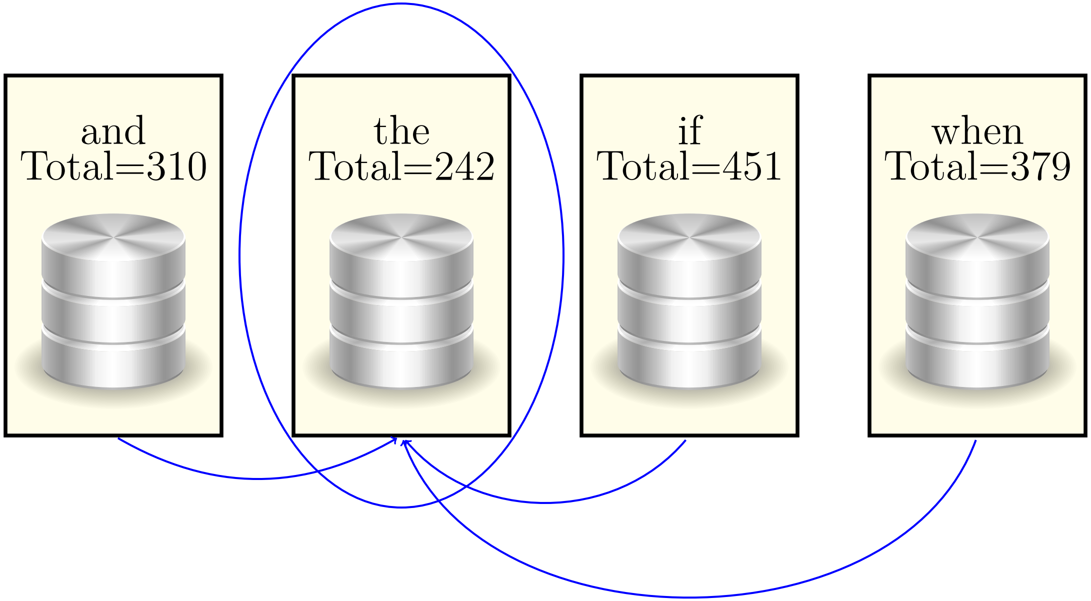
- The main idea is to run multiple copies of each task on carefully chosen servers. Appropriate (coded) data is exchanged among the servers so that the final result can be computed.

[2] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, A Fundamental Tradeoff Between Computation and Communication in Distributed Computing, *IEEE Transactions on Information Theory*, vol. 64, no. 1, pp. 109 128, Jan 2018.

Nature of shuffle traffic

Each node knows all the counts on its assigned chapters

Knows all counts on Ch. 3 & 4



Needs the counts on the word that it reduces

Nature of shuffle traffic

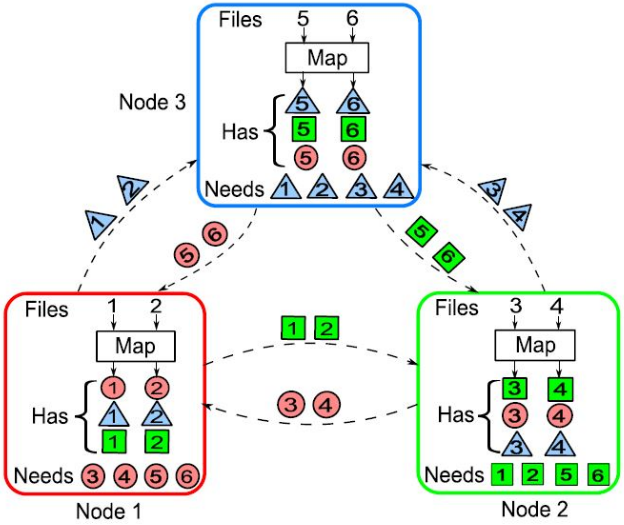
- Can we gain if more than one server processes the same chapter?

Network coding in action!

Shuffle traffic: Uncoded Example

Uncoded scenario

$r = 1$, **# servers = 3, # functions = 3, # subfiles = 6**



[3]

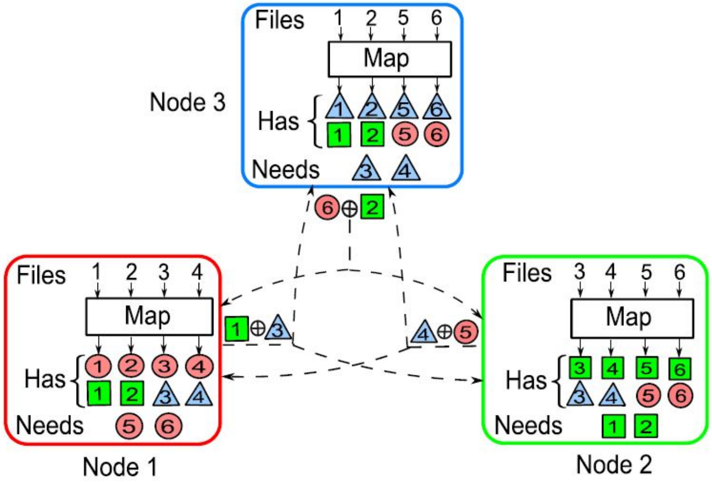
Total of 12 packets tx. over network

[3] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, A Fundamental Tradeoff Between Computation and Communication in Distributed Computing, *IEEE Transactions on Information Theory*, vol. 64, no. 1, pp. 109 128, Jan 2018.

Shuffle traffic: Coded Example

Coded scenario

$r = 2$, # servers = 3, # functions = 3, # subfiles = 6



4

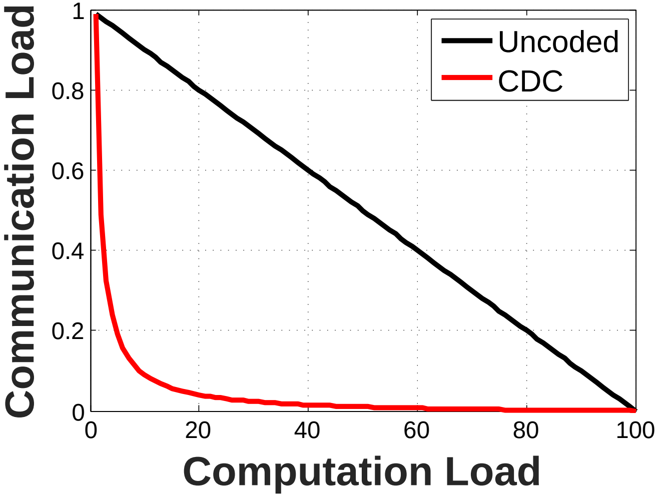
Uncoded approach will require 6 packets tx.

Coded approach requires 3 multicast tx.

[4] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, A Fundamental Tradeoff Between Computation and Communication in Distributed Computing, *IEEE Transactions on Information Theory*, vol. 64, no. 1, pp. 109 128, Jan 2018.

Communication vs. Computation Tradeoff in MapReduce-like systems

Coded Distributed Computing Framework



Coded caching ideas are at the core of this tradeoff ...
“Subpacketize and place (properly)" ...

Formal Problem Statement

- K servers.
- A file split into N input subfiles $w_1, \ldots, w_N$.
- Q functions to be computed $\phi_1, \ldots, \phi_Q$ where ϕ_q is a function of all input subfiles.

$$u_q = \phi_q(w_1, \ldots, w_N)$$

Formal Problem Statement

- Splitting function computation as Map & Reduce.

$$\phi_q(w_1, \ldots, w_N) = h_q(g_{q,1}(w_1), \ldots, g_{q,N}(w_N)),$$

- Map function.

$$\vec{g}_n = (g_{1,n}, \ldots, g_{Q,n}),$$

  maps w_n into Q intermediate values $\nu_{q,n} = g_{q,n}(w_n)$ for $q = 1, \ldots, Q$.

- Reduce function h_q takes $\nu_{q,1}, \ldots, \nu_{q,N}$ as input and completes the computation of ϕ_q.

Map Phase

- Node k computes the Map function on the set of subfiles $\mathcal{M}_k$ stored on it. We assume that no subfile is left unmapped.

Computation Load

The total number of Map functions computed across all the K nodes normalized by the number of files.

$$r \triangleq \frac{\sum_{k=1}^{K} |\mathcal{M}_k|}{N}$$

- Average number of nodes that map a given subfile.

Shuffle Phase

- For simplicity, assume that $Q = K$ and that node k is responsible for reducing function ϕ_k.
- Node k needs the intermediate values that it does not have at the end of the Map phase. This is precisely the set

$$\{\nu_{k,n} : w_n \notin \mathcal{M}_k\}.$$

Shuffle Phase

- Requires the transmission of the messages over the network. For simplicity, assume that each message of size T bits. Suppose that there are $\mathcal{L}_K$ such messages transmitted.

Communication Load

The communication load of the scheme, denoted by L, where $0 \leq L \leq 1$ is defined as

$$L = \frac{\mathcal{L}_K}{QNT}$$

- Represents the normalized number of bits communicated in the Shuffle phase.

Reduce Phase

- Shuffle phase ensures that the i-th node has enough information to reduce $\phi_i(w_1, \ldots, w_N)$, i.e. it has the values $\nu_{i,1}, \ldots, \nu_{i,N}$

- It thus performs the computation

$$h_i(\nu_{i,1}, \ldots, \nu_{i,N}).$$

Formal Result

Computation-Communication Function

$$L^*(r) \triangleq \inf\{L : (r, L) \text{ is feasible}\}$$

Coded Distributed Computing Theorem

The computation-communication function of the CDC framework, $L_{coded}(r)$ is given by

$$L_{coded}(r) = \frac{1}{r}\left(1 - \frac{r}{K}\right), \text{ where } r \in \{1, \ldots, K\},$$

for sufficiently large packets.

As with most things ...

SHANNON WHEELER 2011
"HONESTLY? I PREFERRED WHEN WE DIDN'T TALK ABOUT THE ELEPHANT"

Source: www.appitierre.com

Practical Issues: Subpacketization Level in Coded Caching

- Huge gains of coded caching require each file to be divided into a large number of parts (exponential in problem parameters).
  Subpacketization level F_s
  $$F_s = \binom{K}{KM/N} \approx 2^{KH_2(M/N)}$$

- Example: System with $K = 50$ users, with $M/N = 0.4$, original scheme needs $F_s \approx 10^{14}$.
  - At the bare minimum each file should be of size 100 terabits.
  - In practice, much higher since the basic unit of storage on a computer is a sector.

Practical Issues: Subpacketization Level in Coded Distributed Computing

- Communication load under CDC is

$$L_{coded}(r) = \frac{1}{r} L_{uncoded}(r)$$

where r is the replication factor

- Result requires the number of subfiles to be

$$N = \binom{K}{r} \eta_1, \text{ where } \eta_1 \text{ is a positive integer.}$$

- In practical implementation, small subfiles present rather serious problems.

Related Work

- Problem was first noticed and studied in the context of decentralized coded caching in the work of [Shanmugam et al.].

- For the centralized case, a placement delivery array (PDA) based scheme was proposed in [Yan et al.].
- Prior work by us [Tang-Ramamoorthy] also matches the parameters in [Yan et al.].

Related Work

- A hypergraph perspective introduced by [Shangguan et al.]
  - Demonstrates that (asymptotically) constant rate with subpacketization F_s linear in K is impossible.
  - Shows the existence of schemes where subpacketization scales as $\exp(c\sqrt{K})$.
- Work by [Shanmugam et al.] demonstrates connections with induced matchings in Ruzsa-Szemeredi graphs.
  - Shows the existence of schemes where subpacketization is linear but rate scales as K^δ, where $0 < \delta < 1$.
- Both these results require rather large values of K and are not applicable in a real-life scenarios.

Tackling the subpacketization problems in both these areas …

- Huge gains appear to need exponential scaling (in K) of the number of subfiles N.
- Key issue is that both cases $N = \binom{K}{\alpha}$ where the value α depends on the problem setting
  - $\alpha = KM/N$ for coded caching
  - $\alpha = r$ for coded distributed computing
- Can we do better with fewer subsets with structure ...

An illustrative example

P1	P2	P3
{1, 2}	{1, 3}	{1, 4}
{3, 4}	{2, 4}	{2, 3}

- What can this simple set system do for us?

Usage in a coded caching scenario

- Users correspond to blocks (total of six). System with $K = 6$ users and $N \geq K$.

$$\begin{array}{ll} \{1,2\} \Rightarrow U_{\{1,2\}} & \{3,4\} \Rightarrow U_{\{3,4\}} \\ \{1,3\} \Rightarrow U_{\{1,3\}} & \{2,4\} \Rightarrow U_{\{2,4\}} \\ \{1,4\} \Rightarrow U_{\{1,4\}} & \{2,3\} \Rightarrow U_{\{2,3\}} \end{array}$$

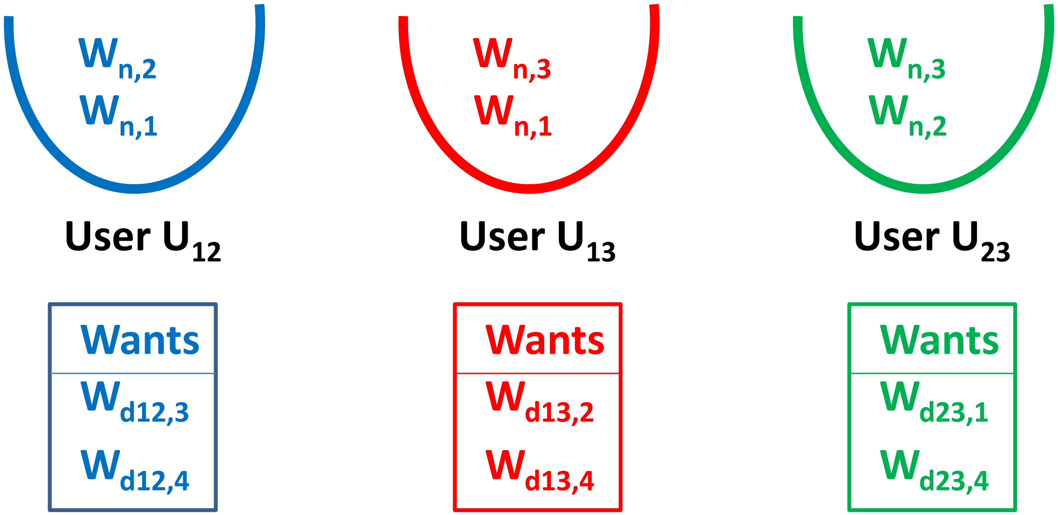
- Subfiles correspond to points (total of four). Split each file into 4 subfiles $W_n = \{W_{n,1}, \ldots, W_{n,4}\}$.
- For original scheme, users correspond to points (K points), subfiles correspond to $\binom{K}{\frac{KM}{N}}$ blocks (all subsets of size $\frac{KM}{N}$ of $\{1, \cdots, K\}$)
- Placement Phase: User $U_{\{i,j\}}$ caches for each file W_n in the library,

$$Z_{i,j} = \{W_{n,i}, W_{n,j}\}$$

  - For example, user $U_{\{1,2\}}$ caches $\{W_{n,1}, W_{n,2}\}_{n=1}^{N}$.
  - Each user thus caches exactly half of each file. Therefore,

$$\frac{M}{\ } = \frac{1}{\ }$$

Example of delivery phase scheme



$W_{n,2}$
$W_{n,1}$

User U_{12}

Wants
$W_{d12,3}$
$W_{d12,4}$

$W_{n,3}$
$W_{n,1}$

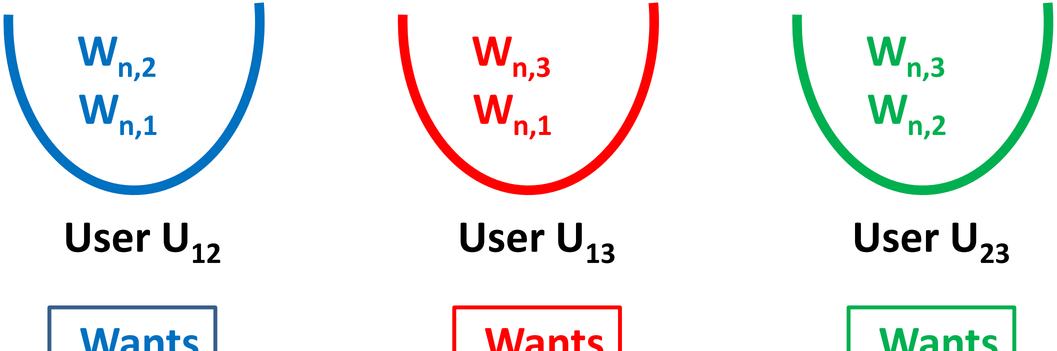
User U_{13}

Wants
$W_{d13,2}$
$W_{d13,4}$

$W_{n,3}$
$W_{n,2}$

User U_{23}

Wants
$W_{d23,1}$
$W_{d23,4}$



$W_{n,2}$
$W_{n,1}$

User U_{12}

Wants

$W_{n,3}$
$W_{n,1}$

User U_{13}

Wants

$W_{n,3}$
$W_{n,2}$

User U_{23}

Wants

Rate Analysis

- Four such equations can be found. Each equation corresponds to $\frac{1}{4}$-th of a file. Total rate is therefore 1 (normalized by file size).
- System has $K = 6$ users, $N \geq K$, $M/N = 1/2$ and a subpacketization level of $F_s = 4$.
- Original scheme would have subpacketization level of $\binom{6}{3} = 20$ and a rate of 0.75.

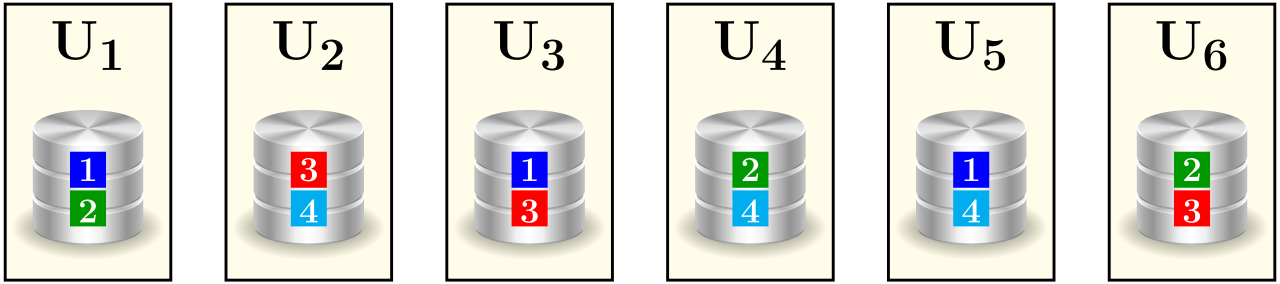
Usage in a Coded Distributed Computing Scenario

The same set system can be used in a Coded Distributed Computing setting!!

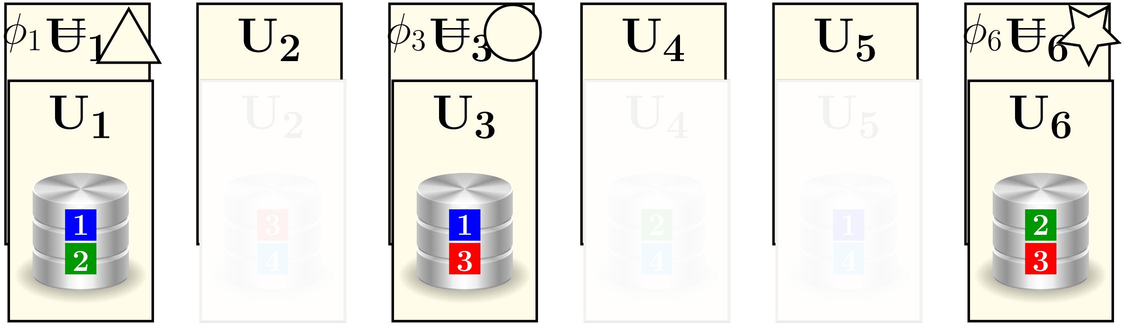
Usage in a Coded Distributed Computing Scenario

- $K = 6,\ Q = 6,\ N = 4,\ r = 3$

  Consider the following Map policy and let the i-th server reduce the i-th function (Reduce policy)



Usage in a Coded Distributed Computing Scenario



Group of servers $G = \{U_1, U_3, U_6\}$

△n $= \phi_1(w_n)$

Each value △n can be split into two parts △n[1] and △n[2].

Server	Knows	Needs	Transmits	Recovers
U_1	$\phi_j(w_1), \phi_j(w_2), \forall j$	$\phi_1(w_n), n = 3, 4$	●2$_{[1]}$ ⊕ ★1$_{[2]}$	▲3
U_3	$\phi_j(w_1), \phi_j(w_3), \forall j$	$\phi_3(w_n), n = 2, 4$	★1$_{[1]}$ ⊕ ▲3$_{[2]}$	●2
U_6	$\phi_j(w_2), \phi_j(w_3), \forall j$	$\phi_6(w_n), n = 1, 4$	●2$_{[2]}$ ⊕ ▲3$_{[1]}$	★1

Server	Knows	Needs	Transmits	Recovers
U_1	$\phi_j(w_1), \phi_j(w_2), \forall j$	$\phi_1(w_n), n = 3, 4$	●2$_{[1]}$ ⊕ ★1$_{[2]}$	▲3
			★1 ⊕ ▲3	

Coded Distributed Computing example

- Choice of groups can be made systematic

Group	Server	Transmits
G_1	U_1	$\nu_{3,2}[1] \oplus \nu_{6,1}[2]$
	U_3	$\nu_{6,1}[1] \oplus \nu_{1,3}[2]$
	U_6	$\nu_{3,2}[2] \oplus \nu_{1,3}[1]$
G_2	U_1	$\nu_{5,2}[1] \oplus \nu_{4,1}[1]$
	U_4	$\nu_{5,2}[2] \oplus \nu_{1,4}[1]$
	U_5	$\nu_{4,1}[2] \oplus \nu_{1,4}[2]$
G_3	U_2	$\nu_{5,3}[1] \oplus \nu_{3,4}[1]$
	U_3	$\nu_{5,3}[2] \oplus \nu_{1,1}[1]$
	U_5	$\nu_{3,4}[2] \oplus \nu_{1,1}[2]$
G_4	U_2	$\nu_{6,4}[1] \oplus \nu_{4,3}[1]$
	U_4	$\nu_{6,4}[2] \oplus \nu_{2,2}[1]$
	U_6	$\nu_{2,2}[2] \oplus \nu_{4,3}[2]$

Rate Analysis

- Each group has three equations each of size $T/2$. Therefore

$$L_{prop} = \frac{4 \cdot 3 \cdot T/2}{N \cdot Q \cdot T} = \frac{1}{4}$$

Addressing the subpacketization bottleneck

Some systematic techniques for addressing the subpacketization issue(s) ..

Placement Delivery Arrays (Coded Caching)

- K-number of server, M - cache size, N - number of files, F - subpacketization level.

Placement Delivery Array (PDA)

A PDA is a $F \times K$ array $\mathbf{P}$ whose entries lie in $\{0, 1, \ldots, S-1\} \cup \{*\}$. Columns correspond to users and rows to subfiles.

- $P_{j,k} = *$ implies that user k caches $W_{i,j}$ for $i = 1, \ldots, N$.
- The other entries indicate the equations to be transmitted in the delivery phase

Simultaneously specifies placement and the delivery phase.

- Some conditions need to be satisfied ...[5]

[5] Q. Yan, M. Cheng, X. Tang and Q. Chen, On the Placement Delivery Array Design for Centralized Coded Caching Scheme, *IEEE Trans. on Info. Th., vol. 63, no. 9, pp. 58215833, Sep. 2017.*

PDA Example

- Same running example can be represented as a PDA

$$\mathbf{P} = \begin{pmatrix} * & 1 & * & 2 & * & 0 \\ 0 & * & * & 3 & 1 & * \\ * & 3 & 0 & * & 2 & * \\ 2 & * & 1 & * & * & 3 \end{pmatrix}$$

$$\mathbf{P} = \begin{pmatrix} * & 1 & * & 2 & * & \mathbf{0} \\ \mathbf{0} & * & * & 3 & 1 & * \\ * & 3 & \mathbf{0} & * & 2 & * \\ 2 & * & 1 & * & * & 3 \end{pmatrix}$$

- Delivery phase equations are coded into $\mathbf{P}$. For instance, in first time slot server transmits

Formal Definition

Placement Delivery Array

A (K, F, Z, S) PDA $\mathbf{P}$ is an array of size $F \times K$ with entries from $\{0, \ldots, S-1\} \cup \{*\}$ such that

C1. Symbol $*$ appears Z times in each column.

C2. Each integer occurs at least once in the array. If each integer occurs exactly g times,then we call it $g-(K, F, Z, S)$ PDA.

C3. Let $P_{j_1,k_1} = P_{j_2,k_2} = s$. Then,

- $j_1 \neq j_2$ and $k_1 \neq k_2$, i.e., the entries are on different rows and columns.
- The 2×2 sub-array induced by rows j_1, j_2 and columns k_1, k_2 looks like

$$\begin{pmatrix} s & * \\ * & s \end{pmatrix} \quad or \quad \begin{pmatrix} * & s \\ s & * \end{pmatrix}.$$

Placement Phase using PDA

- Each of the N files in the server is split into F subfiles.

- User k caches

$$\bigcup_{i=1}^{N}\{W_{i,j} : P_{j,k} = *\}.$$

- Simple calculation yields

$$\frac{Z}{F} = \frac{M}{N}.$$

Delivery Phase using PDA

- Suppose that server receives request

$$\mathbf{d} = (d_0, \ldots, d_{K-1})$$

- Server successively transmits S equations that are specified by the locations of the integers in $\mathbf{P}$. Specifically, for $s \in \{0, \ldots, S-1\}$

$$\bigoplus_{P_{j,k}=s, j\in\{0,\ldots,F-1\}, k\in\{0,\ldots,K-1\}} W_{d_k,j}.$$

Why does the delivery phase work?

- For a given $s \in \{0, \ldots, S-1\}$, the sub-array induced by the rows & columns where s occurs looks like

$$\begin{pmatrix} s & * & \cdots & * \\ * & s & \cdots & * \\ \vdots & \vdots & \ddots & \vdots \\ * & * * \cdots & s & \end{pmatrix}$$

under appropriate row-column permutations.

- The s-th equation transmitted by the server has the “all-but-one" property.

PDA Scheme - Main Results

Lower Bound on Subpacketiztion for PDA schemes

For a $g-(K,F,Z,S)$ PDA $\mathbf{P}$, if $g = KZ/F + 1 \geq 2$, then

$$F \geq \binom{K}{K\frac{Z}{F}}.$$

Achievable Scheme with PDAs

There exists $g-(K,F,Z,S) = (m+1)-(q(m+1), q^m, q^{m-1}, q^{m+1}-q^m)$ PDA $\mathbf{P}$. This results in a coded caching scheme with

$$K = q(m+1),$$
$$\frac{M}{N} = 1/q, \text{ and}$$
$$R = q-1.$$

Can be shown to be exponentially smaller than $\binom{q(m+1)}{m+1}$.

Linear Block Codes & Subpacketization

Linear block codes (under mild conditions) ⟹ Caching schemes and Coded Distributed Schemes[6]

[6] L. Tang and A. Ramamoorthy, Coded caching schemes with reduced subpacketization from linear block codes, *IEEE Trans. on Info. Th., vol. 64, no. 4, pp. 3099 3120, Apr. 2018.*

Designs

Definition

A design is a pair $(X, \mathcal{A})$ such that

1. X is a set of elements called points, and
2. $\mathcal{A}$ is a collection (i.e., multiset) of nonempty subsets of X called blocks, where each block contains the same number of points.

$$X = \{1, 2, 3, 4\} \text{ (Point set)}$$
$$\mathcal{A} = \{\{1, 2\}, \{3, 4\}, \{1, 3\}, \{2, 4\}, \{1, 4\}, \{2, 3\}\} \text{ (Block set)}$$

Resolvable design

Definition

A parallel class $\mathcal{P}$ in a design $(X, \mathcal{A})$ is a subset of disjoint blocks from $\mathcal{A}$ whose union is X. A partition of $\mathcal{A}$ into several parallel classes is called a resolution, and $(X, \mathcal{A})$ is said to be a resolvable design if $\mathcal{A}$ has at least one resolution.

$$X = \{1, 2, 3, 4\} \text{ (Point set)}$$
$$\mathcal{A} = \{\{1, 2\}, \{3, 4\}, \{1, 3\}, \{2, 4\}, \{1, 4\}, \{2, 3\}\} \text{ (Block set)}$$

Each parallel class is a partition of X.

P1	P2	P3
{1, 2}	{1, 3}	{1, 4}
{3, 4}	{2, 4}	{2, 3}

Linear Codes and Resolvable Designs

Let $q = 2$, $k = 3$. Consider a $(3, 2)$ Single Parity Check code over $\mathbb{Z}_2$,

$$\mathbf{G}_{SPC} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \text{(Generator Matrix)}$$

Columns of matrix $\mathbf{T}$ list all $2^2 = 4$ codewords.

$$\mathbf{T} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}.$$

obtained by computing $\mathbf{c} = \mathbf{u} \cdot \mathbf{G}_{SPC}$ for all possible message vectors $\mathbf{u}$.

Codes and Resolvable Designs

- Points correspond to columns $\{1, \ldots, 4\}$.

$$\mathbf{T} = \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

- Blocks are constructed by identifying column index of points in a row.

$$\begin{aligned} [0\ 0\ 1\ 1] \ &\Rightarrow \mathcal{P}_1 : \{1, 2\}, \{3, 4\} \\ [0\ 1\ 0\ 1] \ &\Rightarrow \mathcal{P}_2 : \{1, 3\}, \{2, 4\} \\ [0\ 1\ 1\ 0] \ &\Rightarrow \mathcal{P}_3 : \{1, 4\}, \{2, 3\} \end{aligned}$$

Searching for good codes

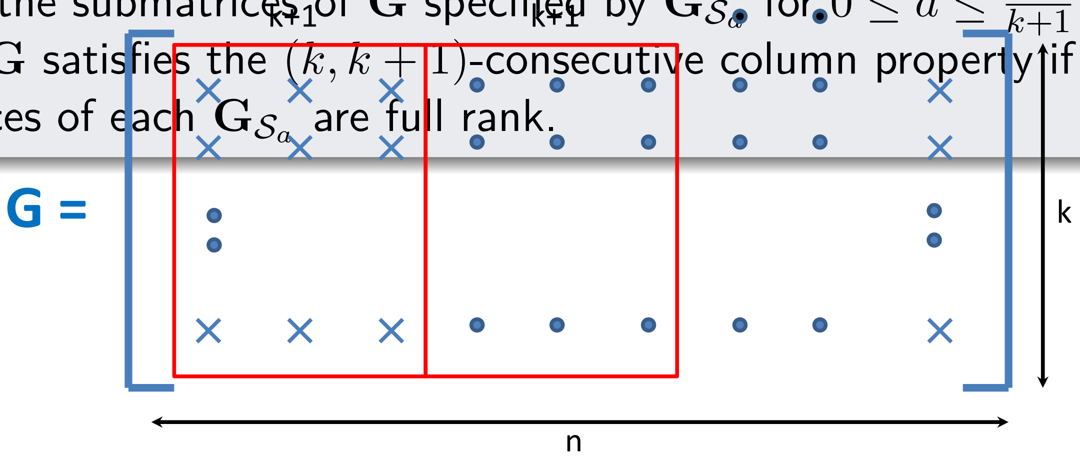
What are the properties of “good" linear block codes for coded caching?

$(k, k+1)$- Consecutive Column Property (CCP)

- Generator matrix $\mathbf{G}$ of size $k \times n$ over $GF(q)$. Let z be the *least* integer such that $k+1 \mid nz$
- Recovery Set: Consider the columns of $\mathbf{G}$ specified by $\mathcal{S}_a$
$$\mathcal{S}_a = \{a(k+1), \cdots, a(k+1)+k\} \mod n.$$

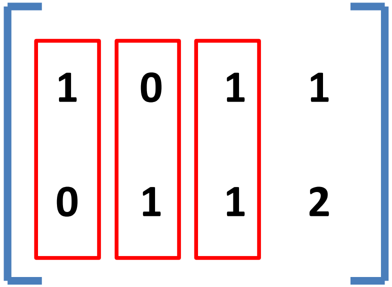
Definition

Consider the submatrices of $\mathbf{G}$ specified by $\mathbf{G}_{\mathcal{S}_a}$ for $0 \leq a \leq \frac{zn}{k+1} - 1$. We say that $\mathbf{G}$ satisfies the $(k, k+1)$-consecutive column property if all $k \times k$ submatrices of each $\mathbf{G}_{\mathcal{S}_a}$ are full rank.



CCP Example - $(4, 2)$ code over $GF(3)$

$\mathcal{S}_0 = \{0, 1, 2\}$



$$\begin{bmatrix} \boxed{\begin{matrix}1\\0\end{matrix}} & \boxed{\begin{matrix}0\\1\end{matrix}} & \boxed{\begin{matrix}1\\1\end{matrix}} & \begin{matrix}1\\2\end{matrix} \end{bmatrix}$$

$$\mathbf{G}_{\mathcal{S}_0} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

All 2×2 submatrices are full-rank.

$\mathcal{S}_1 = \{3, 1, 0\}$

Usage in coded caching scenario

$$\mathbf{T} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & 1 & 1 & 1 & 2 & 2 & 2 \\ 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 \\ 0 & 1 & 2 & 1 & 2 & 0 & 2 & 0 & 1 \\ 0 & 2 & 1 & 1 & 0 & 2 & 2 & 1 & 0 \end{bmatrix}.$$

- Point set

$$X = \{0, 1, 2, 3, 4, 5, 6, 7, 8\}.$$

- Block set (Parallel class)

$$P_0 : \{\{0, 1, 2\}, \{3, 4, 5\}, \{6, 7, 8\}\}$$
$$P_1 : \{\{0, 3, 6\}, \{1, 4, 7\}, \{2, 5, 8\}\}$$
$$P_2 : \{\{0, 5, 7\}, \{1, 3, 8\}, \{2, 4, 6\}\}$$
$$P_3 : \{\{0, 4, 8\}, \{2, 3, 7\}, \{1, 5, 6\}\}$$

Usage in coded caching scenario

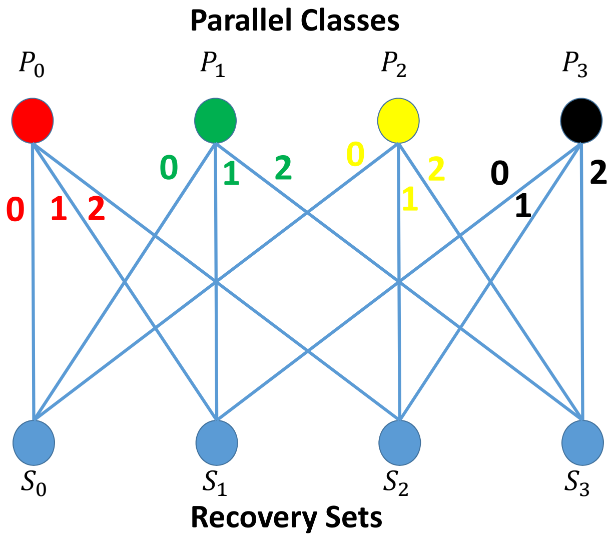
$$\begin{aligned} X &= \{0,1,2,3,4,5,6,7,8\} \\ P_0 &: \{\{0,1,2\},\{3,4,5\},\{6,7,8\}\} \\ P_1 &: \{\{0,3,6\},\{1,4,7\},\{2,5,8\}\} \\ P_2 &: \{\{0,5,7\},\{1,3,8\},\{2,4,6\}\} \\ P_3 &: \{\{0,4,8\},\{2,3,7\},\{1,5,6\}\} \end{aligned}$$

- Users correspond to blocks.
- Subfiles correspond to points and another index $\in \{0,\ldots,z-1\}$.

Example

User U_{012} caches $W^s_{n,0}, W^s_{n,1}, W^s_{n,2}$ for $s = 0,1,2$.

Delivery Phase Algorithm



- Each parallel class $\mathcal{P}_i$ recovers missing subfiles with a specific superscript for each user.
  - Similar to single-parity-check (SPC) code.
- Recovery set bipartite graph formalizes the idea.
- Recovery set $\mathcal{S}_0$: Constructive procedure that picks a user from each parallel class in $\mathcal{P}_0, \mathcal{P}_1$ and $\mathcal{P}_2$.

Delivery Phase Algorithm

- With some work, we can show that the number of user is $K = nq$, cache size is $\frac{M}{N} = \frac{1}{q}$, the rate of our scheme is

  $$R^* = \frac{(q-1)n}{k+1}.$$

  Then coded gain is

  $$g^* = k + 1.$$

- The subpacketization level is

  $$F_s^* = q^k z$$

- Important fact: Both coded gain g^* and subpacketization level F_s^* increase with larger k!!

Delivery Phase Algorithm

- Scheme for $\frac{M}{N} = \frac{1}{q}$ can be converted into a scheme for $\frac{M}{N} = 1 - \frac{k+1}{nq}$ with

$$R^* = \frac{k+1}{(q-1)n}$$

$$F_s^* = (q-1)q^k \frac{zn}{k+1}$$

Some Linear Codes satisfying the CCP

- MDS code over $GF(q)$ of dimension $k \times n$
- Satisfies a stronger condition. *Any $k \times k$ submatrix is full-rank.*
  - Drawback: MDS codes typically need $q + 1 \geq n$.
  - Thus, with increasing n, the cached fraction $M/N = 1/q$ keeps decreasing.
- A large class of cyclic codes!!

$$\mathbf{G} = \begin{bmatrix} g_0 & g_1 & \cdot & \cdot & \cdot & g_{n-k} & 0 & \cdot & \cdot & 0 \\ 0 & g_0 & g_1 & \cdot & \cdot & \cdot & g_{n-k} & 0 & \cdot & 0 \\ \vdots & & & & & & & & & \vdots \\ 0 & 0 & \cdot & 0 & g_0 & g_1 & \cdot & \cdot & \cdot & g_{n-k} \end{bmatrix}$$

Example:$K = nq = 60$ users, $M/N = 1/q = 1/5$

k	F_s	g	R	Notes
-	1.4×10^{12}	13	3.69	Maddah-Ali & Niesen
11	4.9×10^{7}	12	4	
10	-	-	-	$(k, k+1)$-CCP code doesn't exist
9	6.8×10^{6}	10	4.8	
8	1.2×10^{6}	9	5.3	
7	1.5×10^{5}	8	6	
6	10^{5}	7	6.85	
5	3125	6	8	
4	3125	5	9.6	
3	125	4	12	
2	25	3	16	
1	5	2	24	

Table: List of k values and corresponding F_s, g and R

Parameters:$K = nq$ users, $M/N = 1/q$ cache fraction, $F_s = q^k z$ subpacketization level

- Rate comparison (loss is tunable with choice of (n, k) code)

$$\frac{R^*}{R^{MN}} = \frac{n+1}{k+1}$$

- Subpacketization level (exponential improvement)

$$\frac{F_s^*}{F_s^{MN}} \approx q^{k-n} z \left(\frac{q-1}{q}\right)^{n(q-1)}.$$

Parameters: $K = nq$ users, $M/N = 1/q$ cache fraction, $F_s = q^k z$ subpacketization level

- Comparisons when rates of both schemes are almost the same is somewhat harder ...

- Use memory-sharing in Maddah-Ali & Niesen.

- $(9, 5)$ linear block code over $GF(2)$ that satisfies CCP.

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

  Parameters: $K = 9 \times 2 = 18$, $\frac{M}{N} = \frac{1}{2}$, $R^* = \frac{3}{2}$ and $F_s^* = 64$.

- Maddah-Ali & Niesen approach would need memory-sharing between $\frac{M_1}{N_1} = \frac{2}{9}$ and $\frac{M_2}{N_2} = \frac{7}{9}$.
  $R^{MN} \approx 3/2$. However, the subpacketization level is

Linear codes and coded distributed computing

Codewords

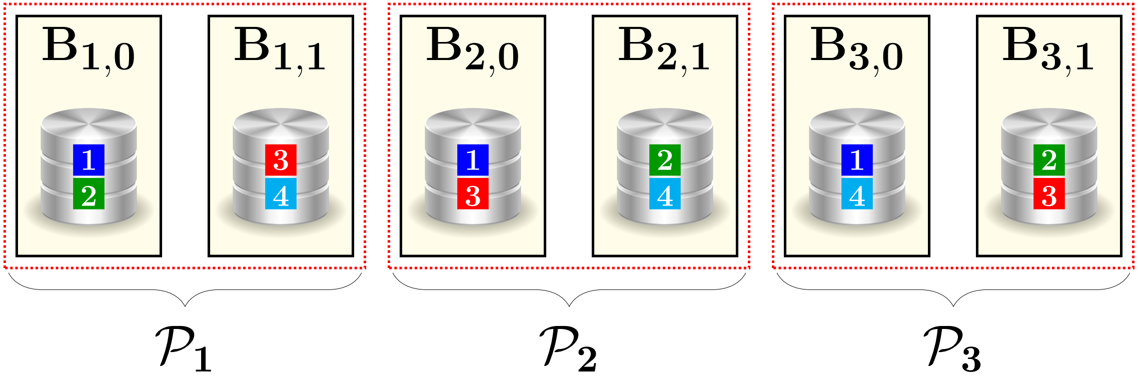
$$\mathbf{T} = \begin{array}{c} \begin{array}{cccc} \mathbf{c}_1^T & \mathbf{c}_2^T & \mathbf{c}_3^T & \mathbf{c}_4^T \end{array} \\ \left[\begin{array}{cccc} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{array}\right] \end{array}$$

Blocks

$$\left.\begin{array}{l} B_{1,0} = \{1,2\} \\ B_{1,1} = \{3,4\} \end{array}\right\} \mathcal{P}_1$$

$$\left.\begin{array}{l} B_{2,0} = \{1,3\} \\ B_{2,1} = \{2,4\} \end{array}\right\} \mathcal{P}_2$$

$$\left.\begin{array}{l} B_{3,0} = \{1,4\} \\ B_{3,1} = \{2,3\} \end{array}\right\} \mathcal{P}_3$$



Placement Phase

- Inputs: File $\mathcal{W}$, Q functions; number of servers $K = k \times q$. K divides Q.
- Split $\mathcal{W}$ into q^{k-1} disjoint subfiles, $w_1, \ldots, w_{q^{k-1}}$.
- Assign subfiles to servers such that server $B_{i,j}$ is assigned subfile w_ℓ if $\ell \in B_{i,j}$

Shuffle phase

- Carried out within groups.
- Form groups of k servers so that every subset of $k-1$ of them shares a file that the remaining server needs.
- Use coded multicast transmission within the group.

Main Results

Communication Load of proposed scheme

The proposed resolvable design based scheme for Coded distributed Computing achieves the following communication load.

$$L_{prop} = \frac{1}{k-1}\left(1 - \frac{k}{K}\right)$$

with a subpacketization level of q^{k-1}. The computation load is $r = k$.

- In contrast, original scheme has $L_{CDC} = \frac{1}{k}\left(1 - \frac{k}{K}\right)$ with a subpacketization of $\binom{kq}{k}$.[7]

[7] K. Konstantinidis and A. Ramamoorthy, Leveraging coding techniques for speeding up distributed computing, *IEEE Global Communications Conference (GLOBECOM), 2018 (to appear).*

Some observations

- Analysis roughly parallels the coded caching setting.
- The Shuffle phase is inspired by, though slightly different from coded caching.
    - In coded caching the server makes all the transmissions.
    - In distributed computing, we form communication groups.

AWS experimental setup

- **TeraSort**

  Widely used sorting benchmark for clusters.

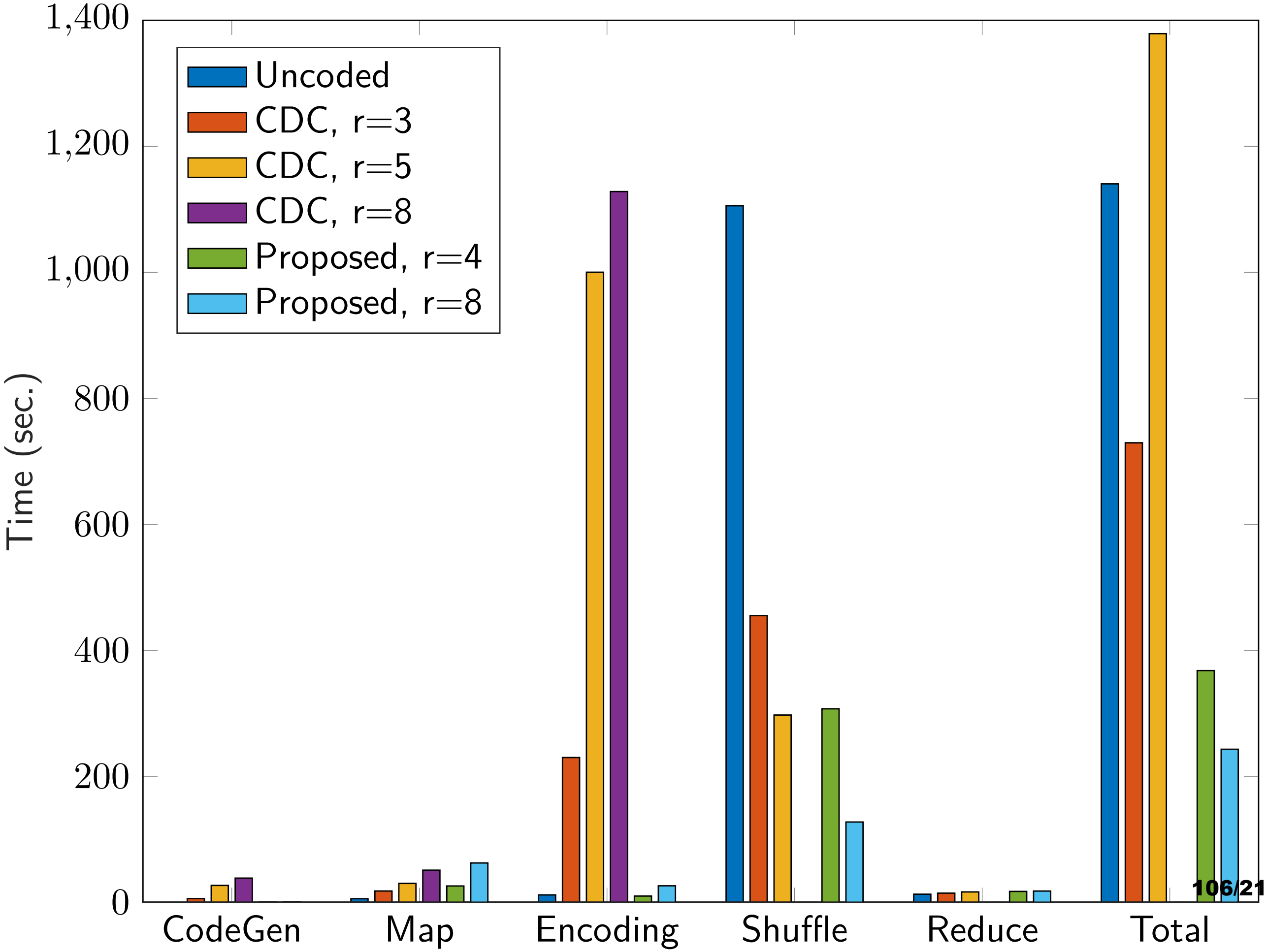
- **Experiment**

  Implemented TeraSort on Amazon Web Services (AWS) cluster.

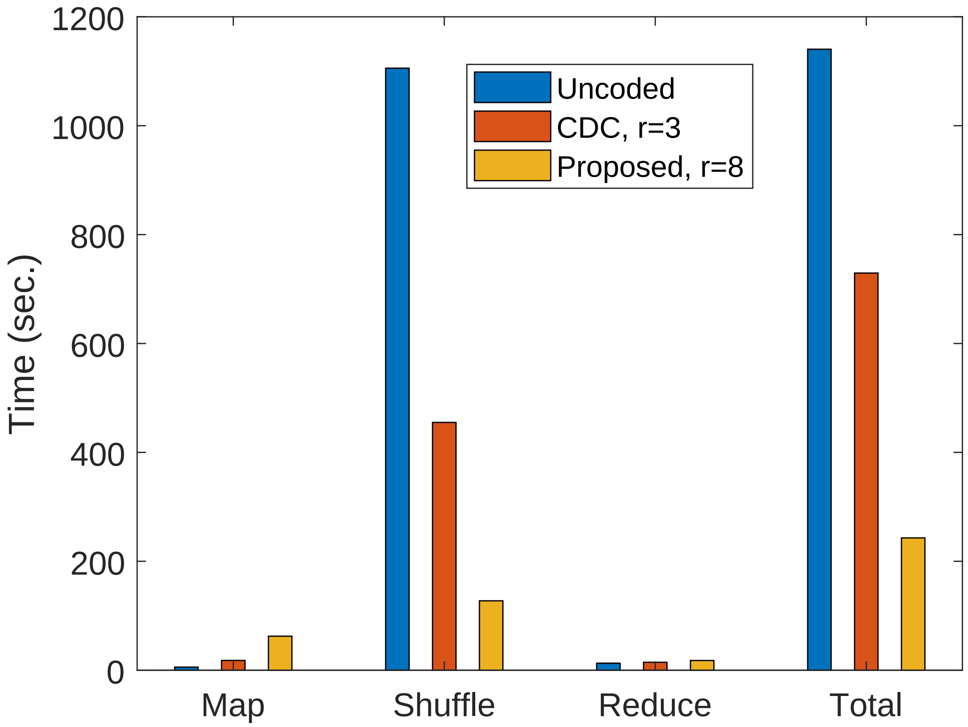
  Data set: 12 GB.

  $K = 16$ servers.

Exhaustive Experimental Results on AWS



Exhaustive Experimental Results on AWS



CDC gain: $\approx 1.56\times$

Proposed gain: $\approx 4.69\times$

Some observations

- Information theoretic load analysis ignores several practical implementation realities that can really affect job execution time.

- Excessive subpacketization can actually cause the execution time to go over the trivial uncoded scheme.

Several Open Questions

- The subpacketization level of most useful constructions still scales exponentially with the problem parameters.
  - Albeit with a much smaller exponent.
  - Provides constructions for tens of users with manageable subpacketization.
  - Open issue: Constructions for general values of K that scale polynomially in problem parameters.
- Constructions that are flexible with respect to the cache fractions that they support.

Is synchronization necessary for coded caching to be useful ....?

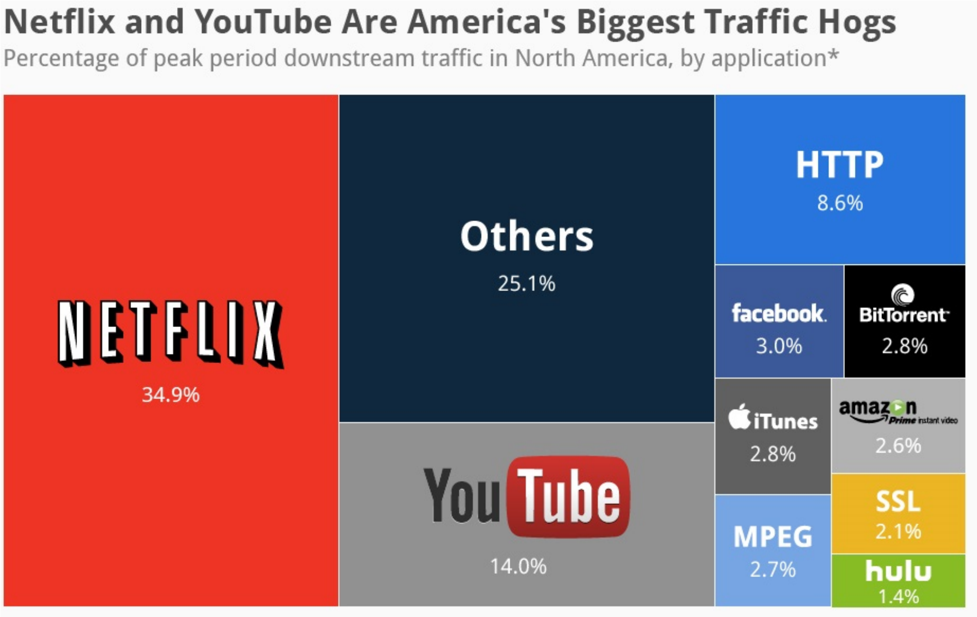
Misconceptions with respect to synchronization

- Huge rate gains assume that the requests of all the users arrive at the same time.
- Users do not have any deadline constraints.

Main Question under consideration

- Effect of coded caching rate under asynchronous request arrival times and hard deadlines.

Motivation



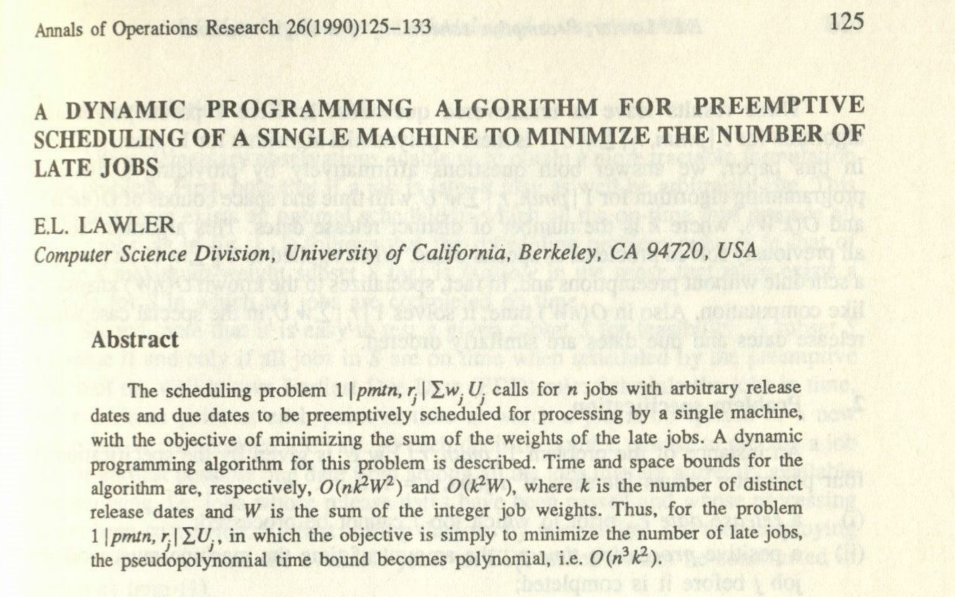
- Different users do not necessarily start watching movies at the same time.
- Users have different levels of delay tolerance.

Related Work

- Considered in the work of [Niesen & Maddah-Ali 2014].
  - Users require different video chunks (part of video).
  - Chunks have deadlines.
  - Algorithm attempts to merge requests to reduce rate of transmission while respecting deadlines.
- To our best knowledge a formal problem setting/analysis that establishes lower bounds on the asynchronous coded caching rate has not been presented.

Related Work: Operations Research

- Problems of this flavor have a rich history in the Operations Research area.



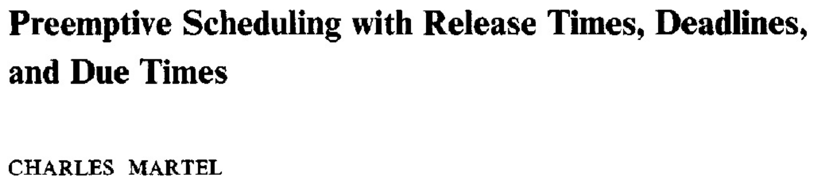
Annals of Operations Research 26(1990)125–133 125

A DYNAMIC PROGRAMMING ALGORITHM FOR PREEMPTIVE SCHEDULING OF A SINGLE MACHINE TO MINIMIZE THE NUMBER OF LATE JOBS

E.L. LAWLER
Computer Science Division, University of California, Berkeley, CA 94720, USA

Abstract

The scheduling problem $1 \mid pmtn, r_j \mid \sum w_j U_j$ calls for n jobs with arbitrary release dates and due dates to be preemptively scheduled for processing by a single machine, with the objective of minimizing the sum of the weights of the late jobs. A dynamic programming algorithm for this problem is described. Time and space bounds for the algorithm are, respectively, $O(nk^2W^2)$ and $O(k^2W)$, where k is the number of distinct release dates and W is the sum of the integer job weights. Thus, for the problem $1 \mid pmtn, r_j \mid \sum U_j$, in which the objective is simply to minimize the number of late jobs, the pseudopolynomial time bound becomes polynomial, i.e. $O(n^3k^2)$.

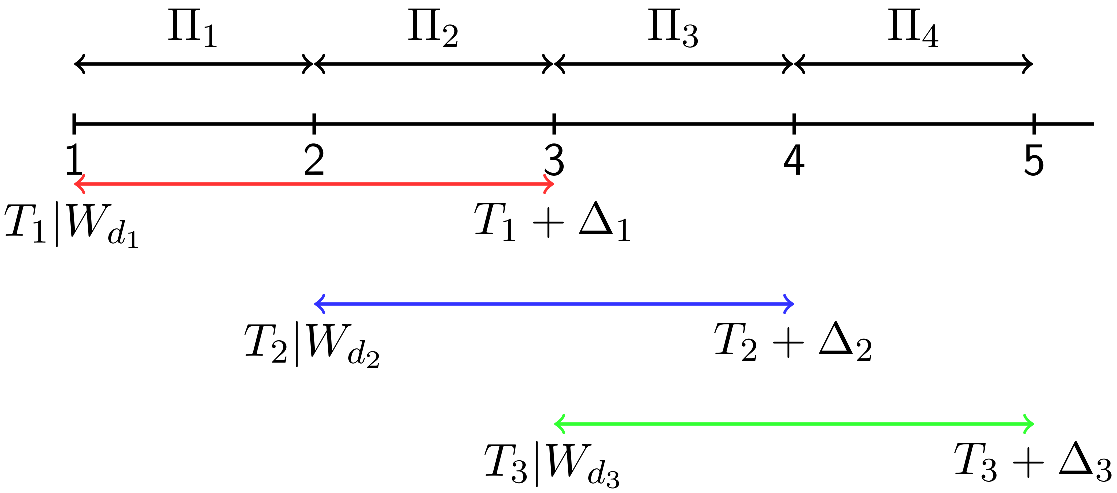


Preemptive Scheduling with Release Times, Deadlines, and Due Times

CHARLES MARTEL

Asynchronous Coded Caching: Problem Formulation

- User requests arrive at different times.
  - T_i: arrival time of the i-th user.
- Each user has a prescribed deadline by which they want their request to be serviced.
  - Δ_i: deadline of the i-th user.
- Packet transmission takes a certain number of time slots.



Placement and delivery schemes

- Assumed a fixed placement scheme and delivery scheme.
- Presented with Maddah-Ali-Niesen placement scheme (approach works for any fixed uncoded placement).
    - Let $t = KM/N$ be an integer and $W_i = \{W_{i,\mathcal{A}} : \ \forall \ \mathcal{A} \subseteq [K], \ |\mathcal{A}| = t\}$ then $Z_i = \{W_{j,\mathcal{A}} : \ i \in \mathcal{A}, \ j \in [K]\}$.
- Delivery scheme: *All-but-one* type delivery signals.

Precise problem statement

- Given the placement scheme and all-but-one type delivery signals.
- Given a sequence of arrival times, deadlines and file requests.
- Determine a schedule of transmissions with minimum rate so that each user is satisfied within their deadline OR determine infeasibility.

Offline

- The server is aware of $\{T_i, \Delta_i, d_i\}_{i=1}^{K}$ for determining schedule.

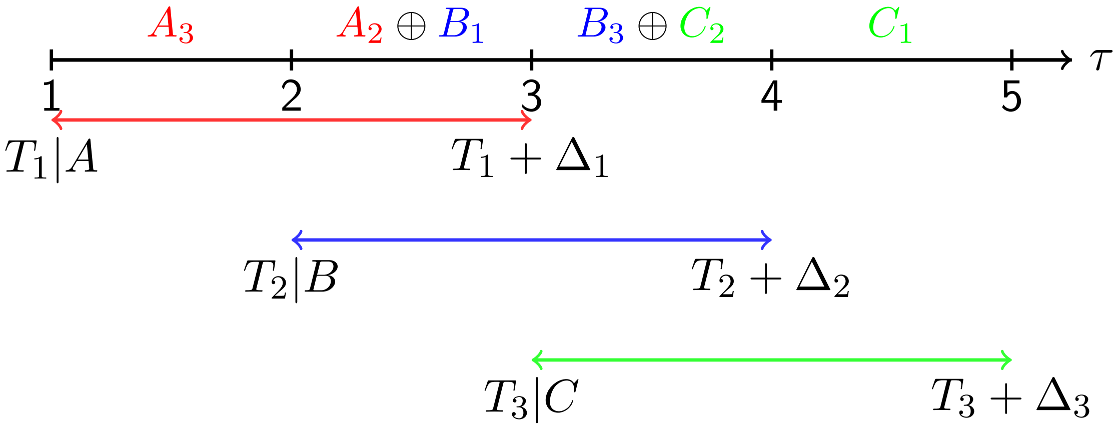
Online

- $\{T_i, \Delta_i, d_i\}$ is revealed to the server as time progress.

- Offline case is a lower bound on the online case.

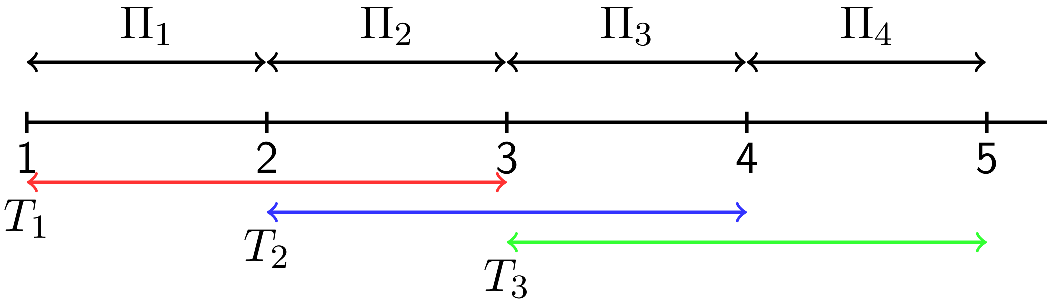
Illustrative Example

System with $N = 3$ files and $K = 3$ users with $M/N = 1/3$.
Placement Scheme: $Z_i = \{A_i, B_i, C_i\}$.



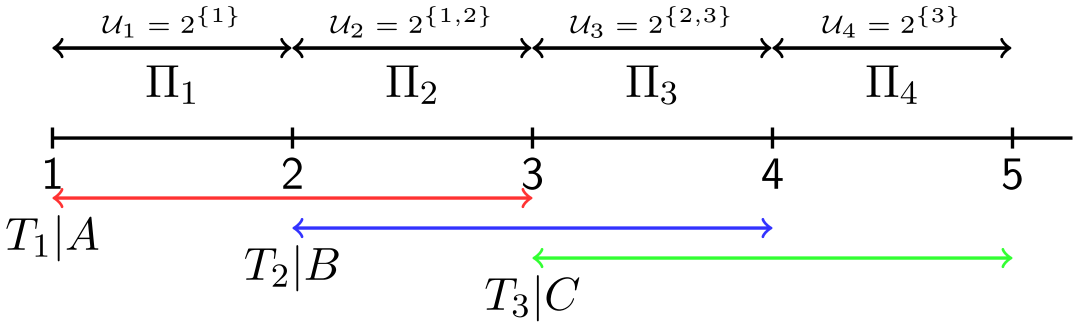
Time intervals

- K arrival times T_i and K deadlines $T_i + \Delta_i$.
- Determine at most $2K - 1$ time intervals.
- Suffices to determine the set of transmitted equations for each interval (rather than each slot).
  - Solution complexity does NOT grow with number of time slots.



User groups

- In each interval, certain user groups can be serviced.
- Let 2^A represent all non-empty subsets of A.


$\mathcal{U}_1 = 2^{\{1\}}$
$\mathcal{U}_2 = 2^{\{1,2\}}$
$\mathcal{U}_3 = 2^{\{2,3\}}$
$\mathcal{U}_4 = 2^{\{3\}}$
Π_1
Π_2
Π_3
Π_4
1
2
3
4
5
$T_1|A$
$T_2|B$
$T_3|C$

User Groups

- Missing subfile $\mathcal{A}$ of user i can be transmitted within user group U if $U \setminus \{i\} \subseteq \mathcal{A}$.

Example

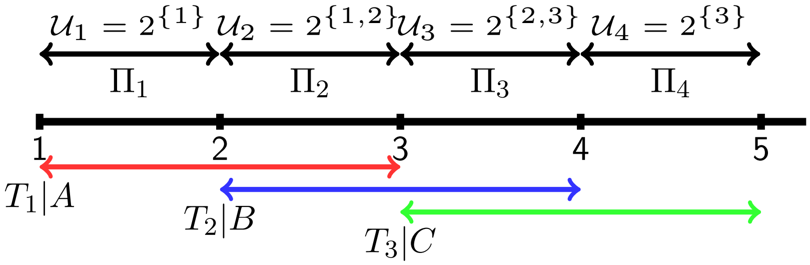
With $N = K = 4$ and $KM/N = t = 2$. Subfile $W_{d_3,\{1,2\}}$ can be transmitted as

$W_{d_3,\{1,2\}}$ (User group $\{3\}$)

$W_{d_3,\{1,2\}} \oplus W_{d_2,\{1,3\}}$ (User group $\{2,3\}$)

$W_{d_3,\{1,2\}} \oplus W_{d_2,\{1,3\}} \oplus W_{d_3,\{1,2\}}$ (User group $\{1,2,3\}$)

Modeling as a multi-commodity flow like problem



$\mathcal{U}_1 = 2^{\{1\}}$ $\mathcal{U}_2 = 2^{\{1,2\}}$ $\mathcal{U}_3 = 2^{\{2,3\}}$ $\mathcal{U}_4 = 2^{\{3\}}$

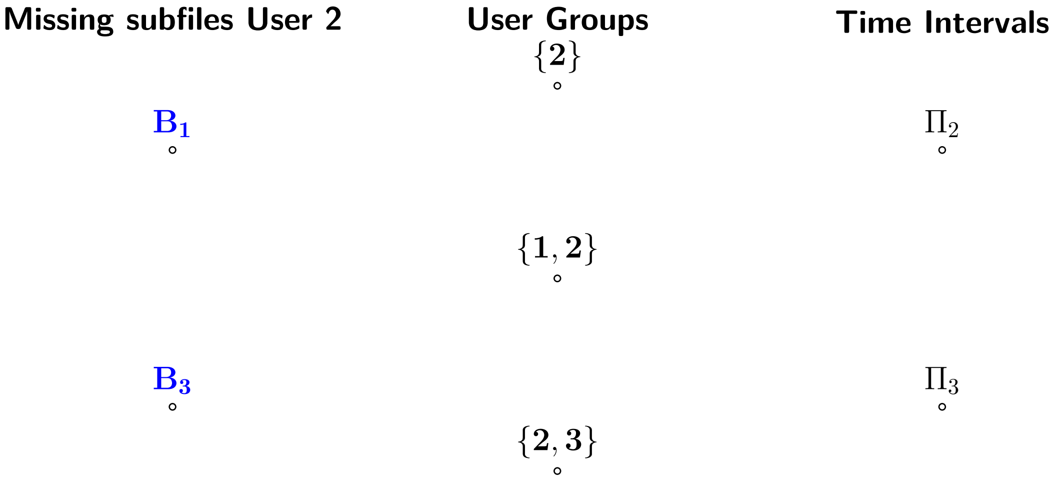
Π_1 Π_2 Π_3 Π_4

1 2 3 4 5

$T_1|A$

$T_2|B$

$T_3|C$



Missing subfiles User 2

$\mathbf{B_1}$

$\mathbf{B_3}$

User Groups

$\{\mathbf{2}\}$

$\{\mathbf{1},\mathbf{2}\}$

$\{\mathbf{2},\mathbf{3}\}$

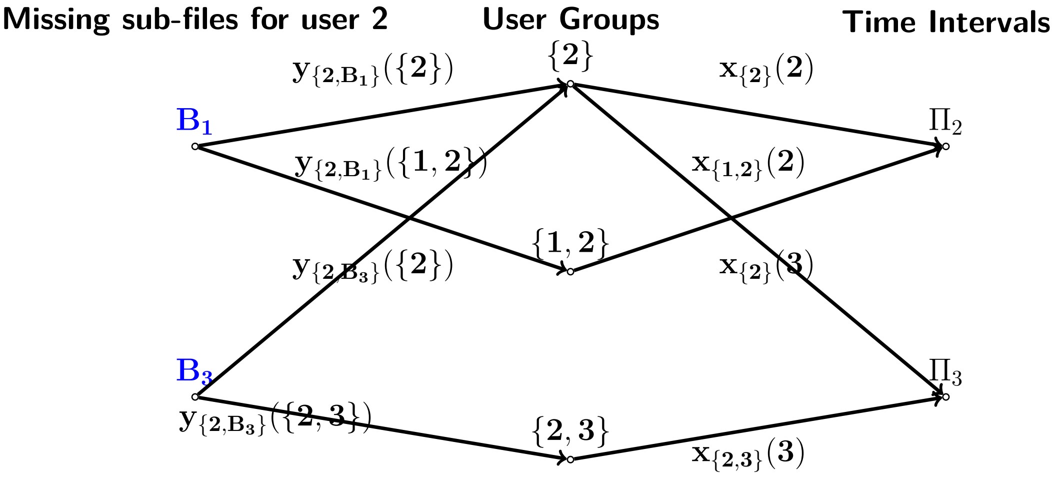
Time Intervals

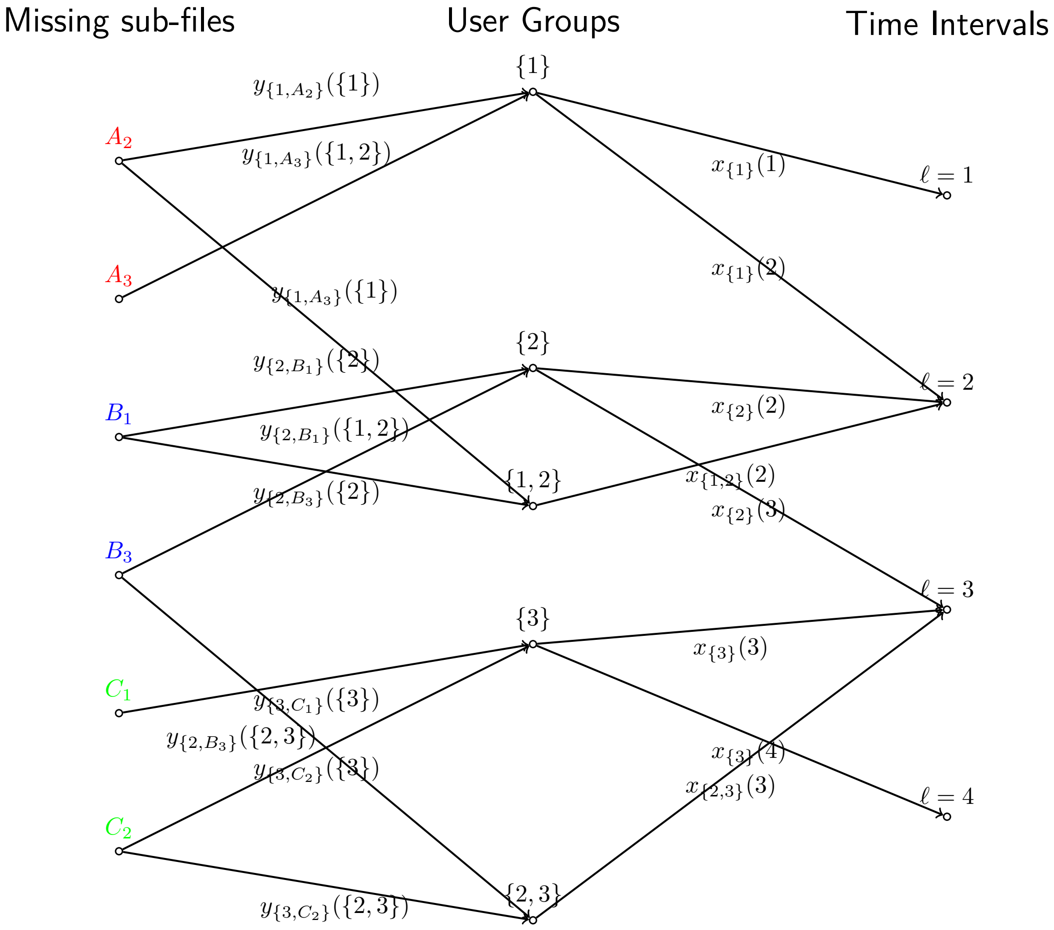
Π_2

Π_3

Modeling as a multi-commodity flow like problem

- $\mathcal{A}$: missing subfile for user i if $i \notin \mathcal{A}$.
- $y_{i,\mathcal{A}}(U)$: portion of subfile $W_{d_i,\mathcal{A}}$ transmitted within user group U for some interval Π_ℓ.
- $x_U(\ell)$: portion of interval Π_ℓ allocated to user group U.


Missing sub-files for user 2
User Groups
Time Intervals
$\{2\}$
$\mathbf{y}_{\{2,B_1\}}(\{2\})$
$\mathbf{x}_{\{2\}}(2)$
B_1
Π_2
$\mathbf{y}_{\{2,B_1\}}(\{1,2\})$
$\mathbf{x}_{\{1,2\}}(2)$
$\{1,2\}$
$\mathbf{y}_{\{2,B_3\}}(\{2\})$
$\mathbf{x}_{\{2\}}(3)$
B_3
Π_3
$\mathbf{y}_{\{2,B_3\}}(\{2,3\})$
$\{2,3\}$
$\mathbf{x}_{\{2,3\}}(3)$


Missing sub-files
User Groups
Time Intervals
A_2
A_3
B_1
B_3
C_1
C_2
$\{1\}$
$\{2\}$
$\{1,2\}$
$\{3\}$
$\{2,3\}$
$\ell = 1$
$\ell = 2$
$\ell = 3$
$\ell = 4$
$y_{\{1,A_2\}}(\{1\})$
$y_{\{1,A_3\}}(\{1,2\})$
$y_{\{1,A_3\}}(\{1\})$
$y_{\{2,B_1\}}(\{2\})$
$y_{\{2,B_1\}}(\{1,2\})$
$y_{\{2,B_3\}}(\{2\})$
$y_{\{3,C_1\}}(\{3\})$
$y_{\{2,B_3\}}(\{2,3\})$
$y_{\{3,C_2\}}(\{3\})$
$y_{\{3,C_2\}}(\{2,3\})$
$x_{\{1\}}(1)$
$x_{\{1\}}(2)$
$x_{\{2\}}(2)$
$x_{\{1,2\}}(2)$
$x_{\{2\}}(3)$
$x_{\{3\}}(3)$
$x_{\{3\}}(4)$
$x_{\{2,3\}}(3)$

Constraints on the flow variables

- All of $W_{d_i,\mathcal{A}}$ needs to be transmitted.

$$\sum_{U\in\mathcal{U}_{\{i,\mathcal{A}\}}} y_{\{i,\mathcal{A}\}}(U) = r \qquad \forall \text{ missing subfiles } \mathcal{A}.$$

$\mathcal{U}_{\{i,\mathcal{A}\}}$: active user groups within deadline of i-th user that can carry $\mathcal{A}$.

- Missing subfiles of user i can share user group U.

$$\sum_{\mathcal{A}\in\mathcal{B}_{\{i,U\}}} y_{\{i,\mathcal{A}\}}(U) \leq \sum_{\ell\in\mathcal{I}_U} x_U(\ell), \quad \forall\, U \in \mathcal{V}_i,\ \forall\, i \in [K]$$

- Π_ℓ is shared between active user groups.

$$\sum_{U\in\mathcal{U}_\ell} x_U(\ell) \leq |\Pi_\ell|$$

where $\mathcal{U}_\ell$ is set of active user groups in time interval Π_ℓ.

LP formulation of the Offline case

$$\min \sum_{\ell=1}^{\beta} \sum_{U \in \mathcal{U}_\ell} x_U(\ell) \tag{1}$$

$$\begin{aligned}
\text{s.t.} \quad & \sum_{U \in \mathcal{U}_\ell} x_U(\ell) \leq |\Pi_\ell|, && \forall\, \ell = 1, \ldots, \beta, \\
& \sum_{\mathcal{A} \in \mathcal{B}_{\{i,U\}}} y_{\{i,\mathcal{A}\}}(U) \leq \sum_{\ell \in \mathcal{I}_U} x_U(\ell), && \forall\, U \in \mathcal{V}_i,\ \forall i \in [K], \\
& \sum_{U \in \mathcal{U}_{\{i,\mathcal{A}\}}} y_{\{i,\mathcal{A}\}}(U) = r, && \forall\, \mathcal{A} \in \Omega^{(i)},\ \forall i \in [K], \\
& 0 \leq x_U(\ell) \leq |\Pi_\ell|, \text{ and} \\
& 0 \leq y_{\{i,\mathcal{A}\}}(U) \leq r.
\end{aligned}$$

8

[8] H. Ghasemi and A. Ramamoorthy, Asynchronous coded caching, *IEEE Intl. Symp. on Information Theory, 2017.*
H. Ghasemi and A. Ramamoorthy, Algorithms for asynchronous coded caching, *Asilomar Conf. on Signals, Systems and Computers, 2017.*

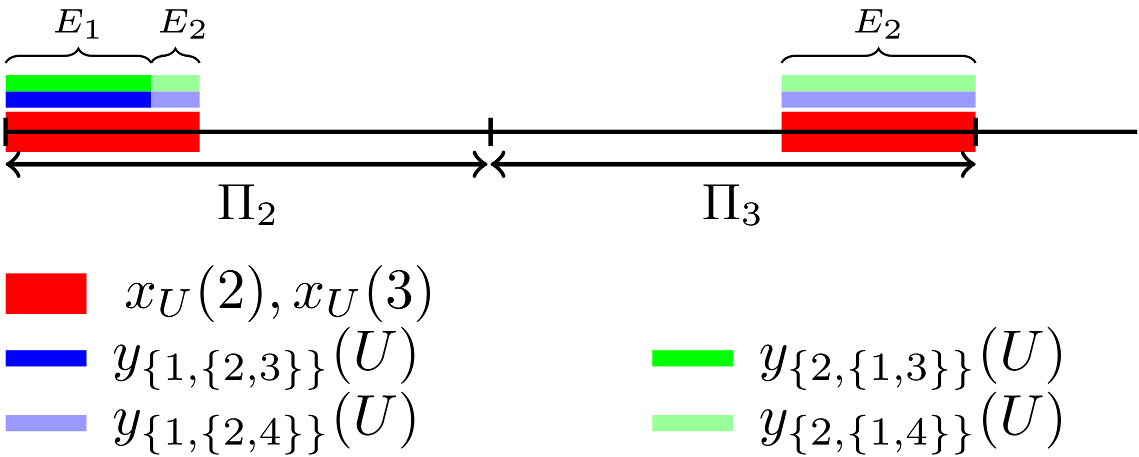
Interpreting LP Solution: Key Idea

- Main point: $y_{i,\mathcal{A}}(U)$ only exists if $U \setminus \{i\} \subseteq \mathcal{A}$.
- $y_{i_1,\mathcal{A}_1}(U), y_{i_2,\mathcal{A}_2}(U), \ldots$ can be superimposed on top of each other. Simple XOR suffices.

Let $U = \{1,2\}$ with $\mathcal{I}_U = \{2,3\}$ we have

$x_U(2) = 0.4$ $\quad$ $x_U(3) = 0.4$

$y_{\{1,\{2,3\}\}}(U) = 0.3 \quad y_{\{2,\{1,3\}\}}(U) = 0.3 \quad y_{\{1,\{2,4\}\}}(U) = 0.5 \quad y_{\{2,\{1,4\}\}}(U) = 0.5$



$E_1 = W_{d_1,\{2,3\}} \oplus W_{d_2,\{1,3\}}$

Dealing with LP complexity

- Number of variables:

$$\leq \{K\binom{K-1}{t} + 2K - 1\}\sum_{j=0}^{t+1}\binom{K}{j}.$$

- Number of constraints:

$$\leq 2K - 1 + (5K-2)\sum_{i=1}^{t+1}\binom{K}{i} + K\binom{K-1}{t}(2\sum_{j=0}^{t+1}\binom{K}{j} + 1).$$

- Usage of time intervals, user groups and interpretation of LP solution reduces complexity substantially with respect to a naive formulation.
  - Still the LP size is large ...
- LP structure is reminiscent of the max-of-flows structure in network coding.
- Can use dual decomposition followed by iterative primal recovery to significantly reduce the time-complexity of determining the schedule.
  - Leverage fast $s-t$ minimum cost network flow algorithms.
  - Simple cycle-Canceling, Minimum-mean cycle-canceling,

Dual Decomposition

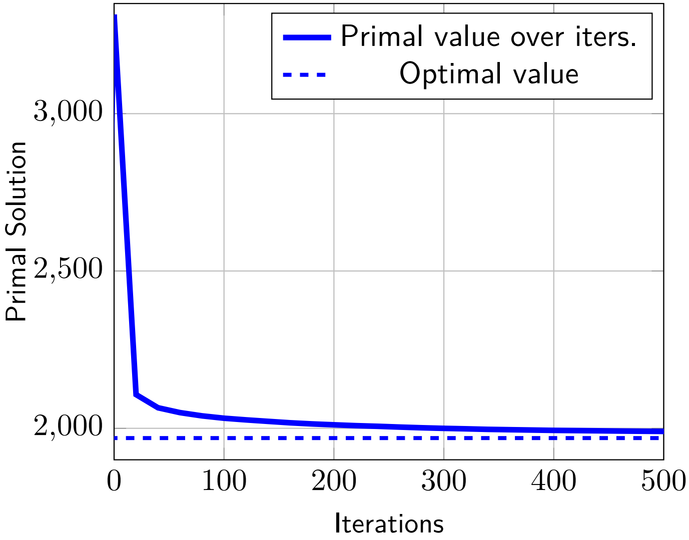
Original LP

$$
\begin{aligned}
\min \sum_{\ell=1}^{\beta} \sum_{U \in \mathcal{U}_\ell} & x_U(\ell) \\
\text{s.t.} \sum_{U \in \mathcal{U}_\ell} & x_U(\ell) \leq |\Pi_\ell|, && \forall\ \ell = 1, \ldots, \beta, \\
\sum_{\mathcal{A} \in \mathcal{B}_{\{i,U\}}} & y_{\{i,\mathcal{A}\}}(U) \leq \sum_{\ell \in \mathcal{I}_U} x_U(\ell), && \forall\ U \in \mathcal{V}_i,\ \forall i \in [K], \\
\sum_{U \in \mathcal{U}_{\{i,\mathcal{A}\}}} & y_{\{i,\mathcal{A}\}}(U) = r, && \forall\ \mathcal{A} \in \Omega^{(i)},\ \forall i \in [K], \\
& 0 \leq x_U(\ell) \leq |\Pi_\ell|, \text{ and} \\
& 0 \leq y_{\{i,\mathcal{A}\}}(U) \leq r.
\end{aligned}
$$

- Recall that $\sum_{\mathcal{A} \in \mathcal{B}_{\{i,U\}}} y_{\{i,\mathcal{A}\}}(U)$ represents the total amount of missing subfiles of user i that are supported by user group U.
- Main idea: Introduce a variable $x_U^{(i)}(\ell)$ such that

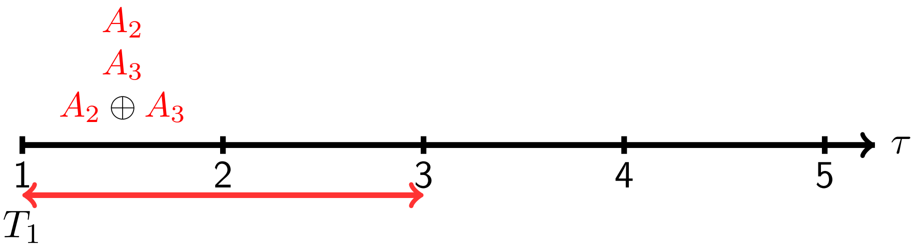
$$\sum y_{\{i,\mathcal{A}\}}(U) = \sum x^{(i)}(\ell)$$

Simulation Results: $N = K = 20, M = 2$

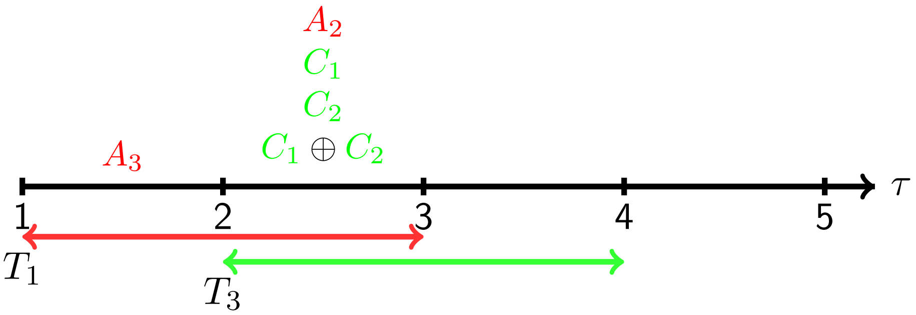


- Dual decomposition based solution $\approx$ 14 seconds.
- Original LP solution takes over 10 minutes.

Online Case: Necessity of coding within missing subfiles of the same user



Suppose, server transmits A_3 in the first time slot.



No way to satisfy first and second arrivals within deadline!

Suppose, server transmits A_2 in first time slot.

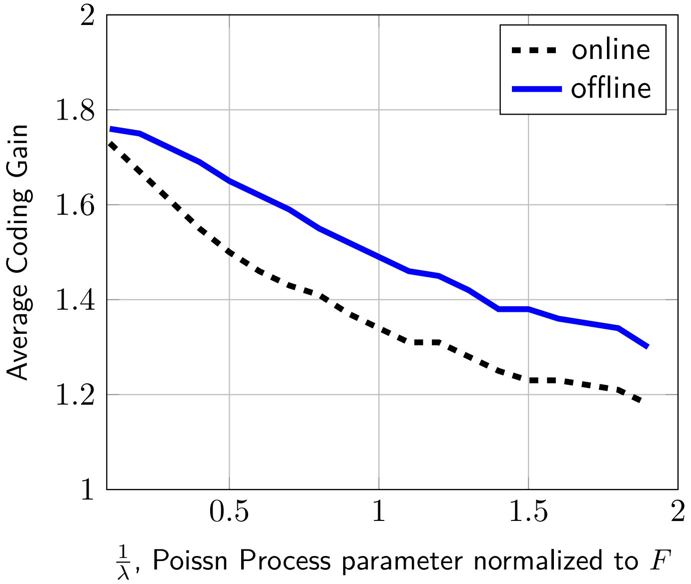
Online Case: Minimization of overall completion time

- Suppose that users do not have strict deadlines but want to minimize the overall completion time with minimum rate of transmission from the server.
- With a little work, we can show that the online rate can be as good as the offline rate.
  - Wait until $t+1$ requests come.
  - Greedily transmit equations that benefit $t+1$ users whenever possible.

An online algorithm inspired by offline LP

- In more recent work, we have developed an online algorithm that solves a new LP whenever a new user makes a request.
- The transmissions use coding over subfiles of the same user.
- The decision to transmit at a certain time depends on a threshold γ, which is calculated based on the severity of the deadlines and the number of users served by the equation.
- The offline LP serves as the limit of what is achievable based on *all-but-one* equations.

Simulation Results: $K = N = 8, M = 2$



- Decentralized placement with $F = 20$.
- Arrivals generated by a Poisson process with parameter λ and deadlines uniformly from a certain interval.
- Graceful degradation of coded caching gain in the presence of asynchronism.

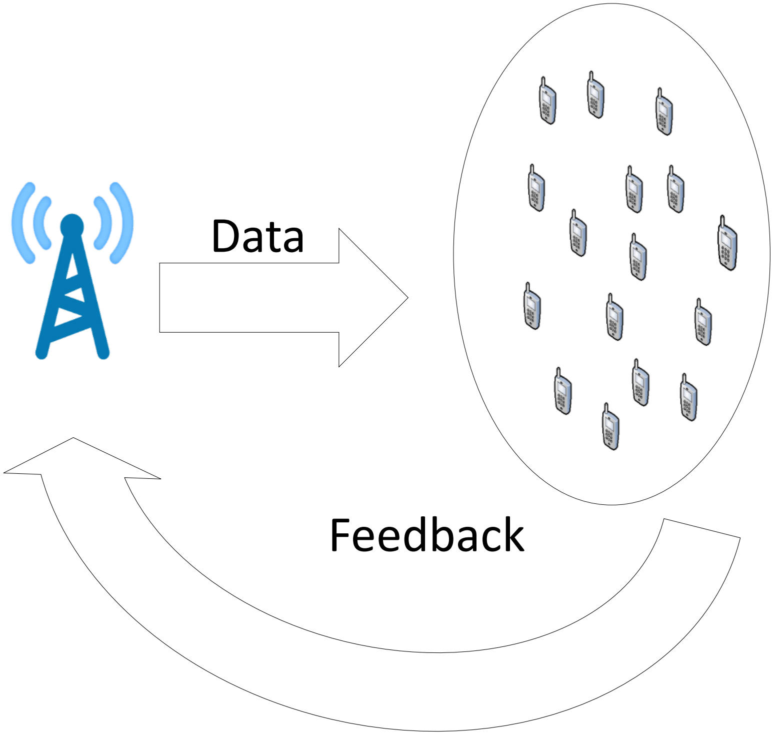
Summary of Asynchronous coded caching

- Presented a formulation of the asynchronous coded caching problem.
- Offline scenario: Can be posed as an LP. Serves as a lower bound on the performance of *any* scheme.
  - Interpretation of the LP as a coding solution.
  - Presented a dual decomposition based fast method for solving the LP. Can solve very large instances ($K = 100$ has been tested).
- Online scenario: Demonstrated the necessity of coding within subfiles intended for the same user. Not required in the synchronized case.
- Several open issues in the online case. Bounding the gap between online vs. offline?

Motivation

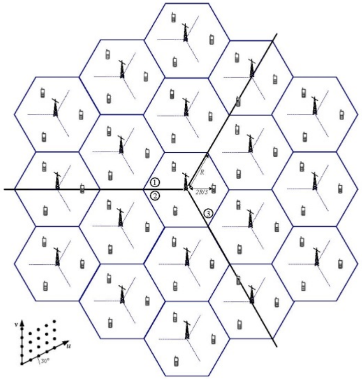
Limitations of Current Communications Paradigms

Main Challenges



- `Feedback': channel-state information (CSI)
  - The instantaneous strengths of each propagation-path between different nodes
- As K increases, the overhead consumes more and more resources
  - No room for actual data
  - Brings current systems and envisioned methods to a halt

Main Challenges: Multi-cell Cooperation

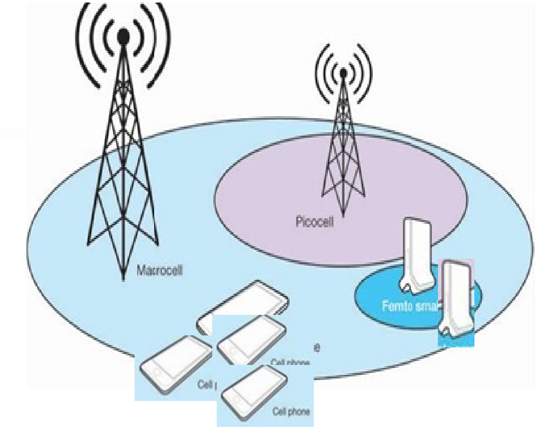


- Even full BS cooperation cannot handle interference
- Spectral efficiency independent of the transmit power
- Cooperation possible only within clusters of limited size (due to CSI)
  - ➢ subject to out-of-cluster interference with power similar to in-cluster signals

$$DOF \stackrel{\text{def}}{=} \frac{Per\ User\ Capacity}{\log SNR} \rightarrow 0 \ \ \text{(as } K \text{ increases)}$$

source: Lozano et al. (2012)

Challenges: Wireless Network Densification



Deployment of more base-stations/access-points per unit area

- Short-range wireless channels different from classical cellular counterparts
  - Exhibit <u>path loss subduction</u> (reduced path loss exponent)
  - <u>Extreme fading</u> (more severe deep-fades)
- SINR decrease after certain densification threshold
- Similar trends are observed for the throughput

⇒ Disruption of densification gains

source: Andrews et al. (2015)

Novel Use of Storage

- This tutorial encourages a novel use of storage in communication and computing networks

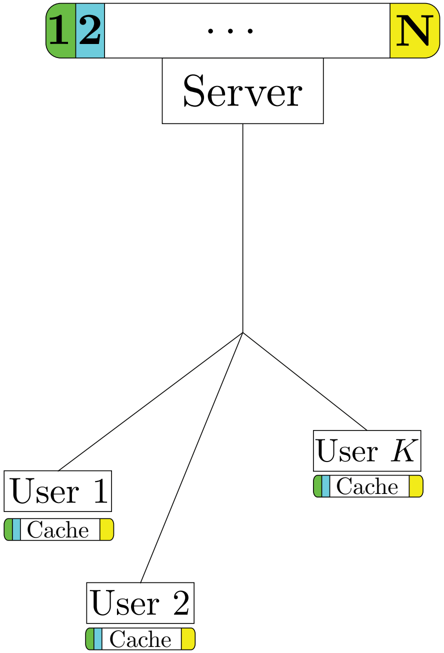
- Using on-board memory at the communicating/computing nodes:
  - NOT to reduce the volume/size of the problem
    - "Prefetch/compute today so that you don't have to tomorrow"
  - BUT to surgically alter the structure of networks
    - Use memory to change networks to something faster and simpler.

Challenge – Modest Cache Sizes

Challenge

Modest Sizes for Caches
Close to “the Edge”

Bottleneck: Caches (γ) are generally small



[Maddah-Ali, Niesen '12]

Setting

- K Users
- $\gamma \in (0,1)$ fractional cache

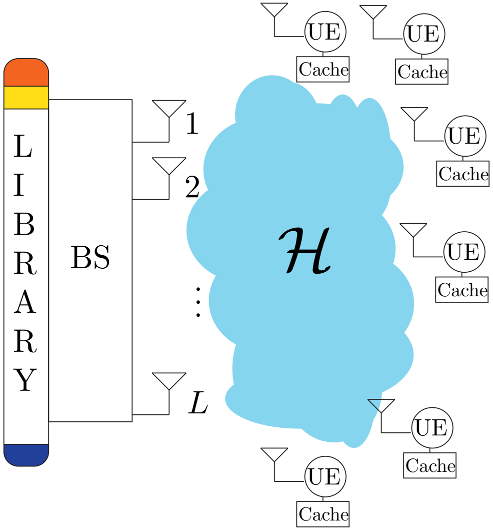
Users Served Simultaneously

$$d_{\Sigma} = K\gamma + 1$$

Big bottleneck: small γ

- Close to 'the edge', γ is expected to be small.
- $\gamma \approx 10^{-3} \rightarrow 10^{-2}$ (ElAyoubi-Roberts, ICN 2015)
- Boost Rx-side caching?!

Coded Caching Using Multiple Antennas



Setting

- K users
- γ fractional cache size
- L antennas

Users served per time-slot

$$d_{\Sigma} = K\gamma + L$$

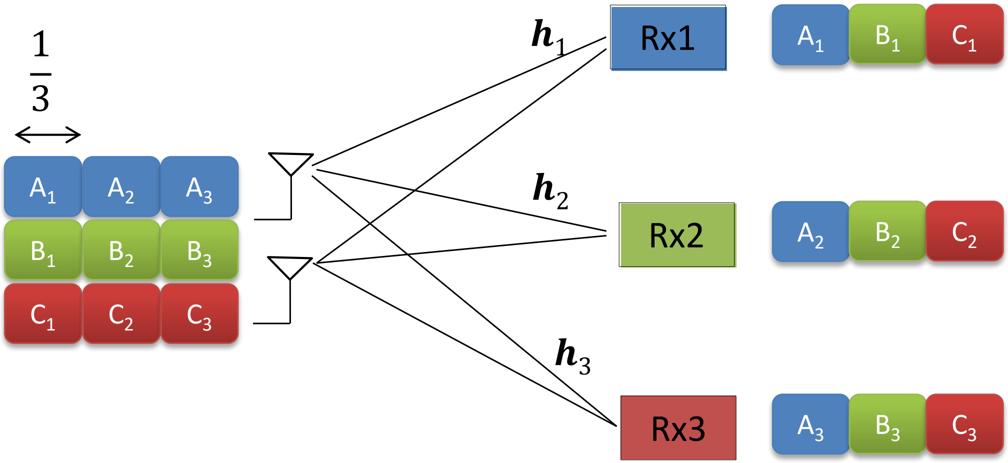
[Shariatpanahi, Motahari & Khalaj, ’15]

Remark

Multiplexing and multicasting gains combine additively.

[Naderializadeh, Maddah-Ali & Avestimehr, ‘16]

Multi-Server Example: $N = K = 3, L = 2, M = 1$



- Setting: MISO-BC (full connectivity) with perfect CSI
- "XORs" of size $K\gamma = 2$

- Find efficient way to deliver these multicasting messages

$$(A_2, B_1); (A_3, C_1);(B_3, C_2)$$

Algorithm: Shariatpanahi et al. (2016)

Multi-Server Example: $N = K = 3, L = 2, M = 1$

- Precode with perfect channel estimates:

$$\boldsymbol{x}_1 = \boldsymbol{h}_1^{\perp} L_1^1(C_2, B_3) + \boldsymbol{h}_2^{\perp} L_2^1(A_3, C_1) + \boldsymbol{h}_3^{\perp} L_3^1(A_2, B_1)$$
$$\boldsymbol{x}_2 = \boldsymbol{h}_1^{\perp} L_1^2(C_2, B_3) + \boldsymbol{h}_2^{\perp} L_2^2(A_3, C_1) + \boldsymbol{h}_3^{\perp} L_3^2(A_2, B_1)$$

where $L(m, n)$ are different linear combinations.

- Take Rx1 for example:

$$y_1 = \boldsymbol{h}_1 \boldsymbol{x}_1 = \boldsymbol{h}_1 \boldsymbol{h}_2^{\perp} L_2^1(A_3, C_1) + \boldsymbol{h}_1 \boldsymbol{h}_3^{\perp} L_3^1(A_2, B_1)$$
$$y_2 = \boldsymbol{h}_1 \boldsymbol{x}_2 = \boldsymbol{h}_1 \boldsymbol{h}_2^{\perp} L_2^2(A_3, C_1) + \boldsymbol{h}_1 \boldsymbol{h}_3^{\perp} L_3^2(A_2, B_1)$$

  - By removing C_1 and B_1 from the cache of receiver 1, A_3 and A_2 can be solved from y_1 and y_2

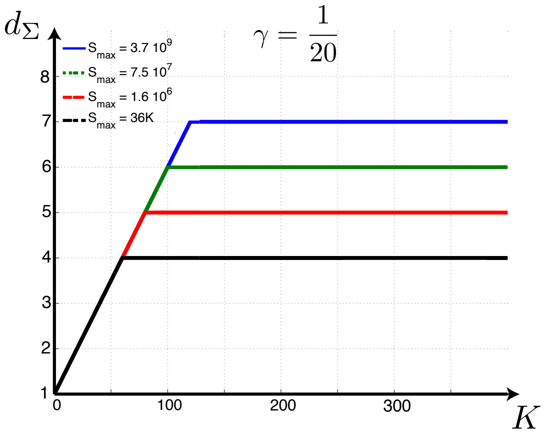
- The rate : each subfile: $|A_3| = \frac{1}{3}$. Sending $\boldsymbol{x}_1, \boldsymbol{x}_2$, takes $\frac{1}{3}$ each

$$T = \frac{2}{3}$$

Algorithm: Shariatpanahi et al. (2016)

TIEDEMANN

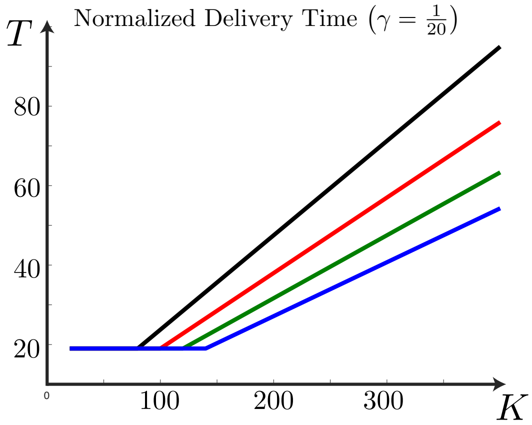
Subpacketization Ramifications



Users Served : 7

File size : ~ 3.7TB

[min packet size 1KB].



Users Served : 7

File size : ~ 3.7 TB

[min packet size 1KB].

Remark

No known setting or algorithm could achieve $G > 5$, if $\gamma \leq \frac{1}{20}$ and $S_{\max} = 10^5$.

[achievability: Shangguan et al. '16]

Subpacketization Bottleneck - Example

State of Art

$K = 50,\ \gamma = 3/10$

$$S_1 = \binom{50}{15} > 10^{12}$$

$$d_\Sigma = 16$$

$$S_{L=5} = \binom{35}{4}\binom{50}{15} > 10^{17}$$

$$d_\Sigma = 20$$

Proposed Solution

New Method

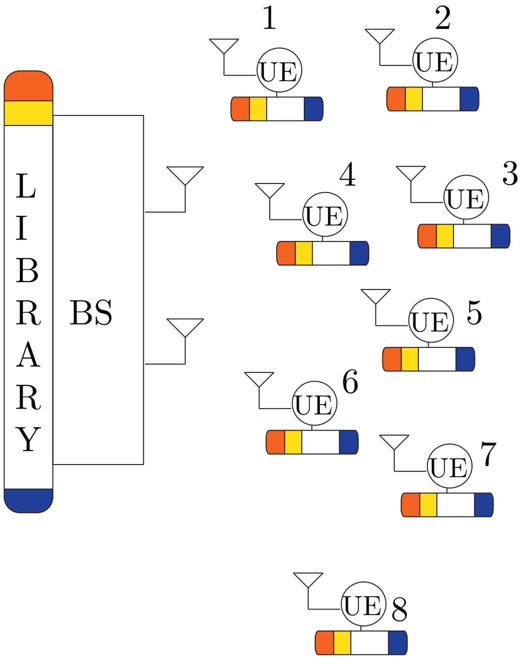
$$S = \binom{50/5}{15/5} = 120$$

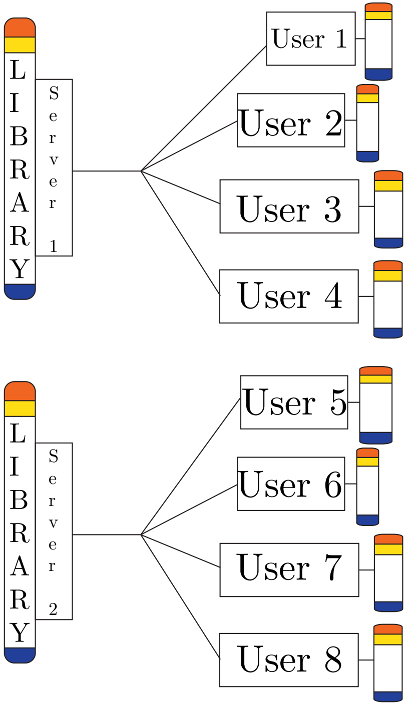
$$d_\Sigma = 20$$

Algorithm Via Example - Wishful Decomposition

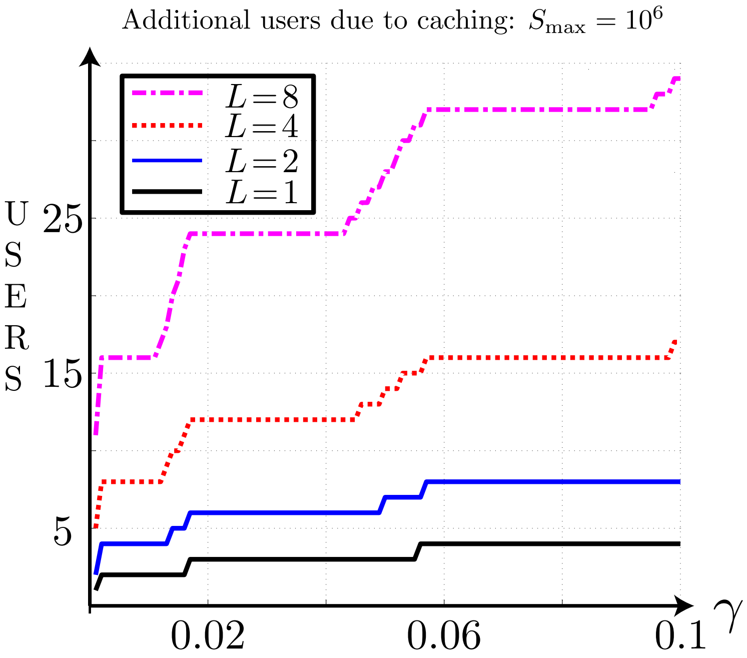
Reality $S \geq \binom{K}{K\gamma}$

Wishful Thinking $S = \binom{K/2}{K\gamma/2}$





Gains



Users Served

$$G = \min\{L \cdot \bar{d}_1, L + K\gamma\}.$$

Subpacketization for unbounded file size

$$S_L = \binom{K/L}{K\gamma/L} \approx \sqrt[L]{\binom{K}{K\gamma}}.$$

[$\bar{d}_1$: CC gain under supacketization constraint in Single Stream setting]

Remarks - Practical Impact of Caching

Perfect World Scenario

∞-File Size

Adding Antennas

$$1 + 28 = 29$$
$$2 + 28 = 30$$
$$3 + 28 = 31$$
$$4 + 28 = 32$$
$$\vdots$$

Harsh Reality

Limited File Sizes

Adding Antennas

$$1 + 7 \to 8$$
$$2 + 7 \to 16$$
$$3 + 7 \to 24$$
$$4 + 7 \to 32$$
$$\vdots$$

“One Bad Apple Spoils the Bunch”

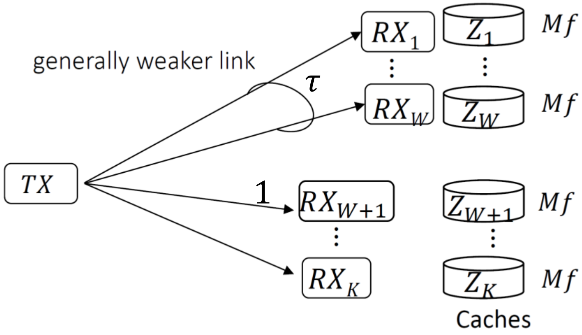
“木桶效应”

Topology Bottleneck of Coded Caching

Features/Opportunities:

- Topological `holes’ to attenuate interference
- XORING on the air

Coded Caching with Uneven Link Strengths

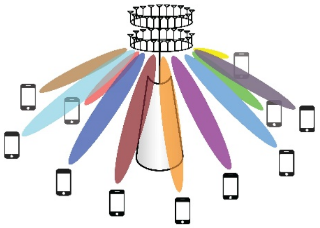


Topologically-uneven wireless SISO K-user BC:

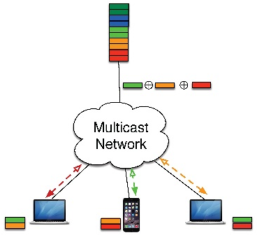
- W weak users with normalized capacity $\tau < 1$
- $K - W$ strong users with normalized capacity $= 1$

- Same cache size per user (M)

- Multiple tx-antennas to reduce channel non-homogeneity



Massive MIMO



Coded caching

- Setting: iid quasi-static Rayleigh fading
- Using n_t transmit antennas to get ‘spatial diversity’
  - Isotropic Multicasting
  - No feedback
- Multicast channel hardens after $\approx logK$ transmit antennas

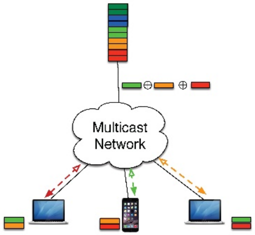
$$n_t \geq \ln(K) + O(1)$$

$P = o(1)$	$\bar{R}_0 = \Theta(P) = o(1)$
$P = \Theta(1)$	$\bar{R}_0 = \Theta(1)$
$P \to \infty$	$\bar{R}_0 \sim \ln(1+P)$

“Scalable wireless content delivery with coded caching” Ngo-Kobayashi-Yang 2017

- Multiple tx-antennas to reduce channel non-homogeneity

Massive MIMO



Coded caching

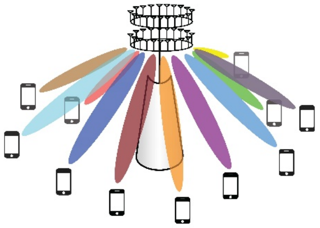
TEMPORAL DIVERSITY ALSO HELPS (no feedback)

If $Ln_t \geq \ln(K) + O(1)$ *and* P *is fixed, then* $\bar{R}_0 = \Theta(1)$ *when* $K \to \infty$.

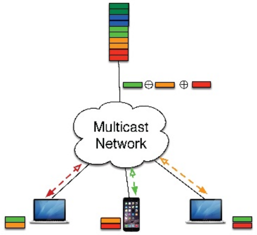
What matters is the total diversity:
space, time, or frequency

“Scalable wireless content delivery with coded caching” Ngo-Kobayashi-Yang 2017

- Multiple tx-antennas to reduce channel non-homogeneity



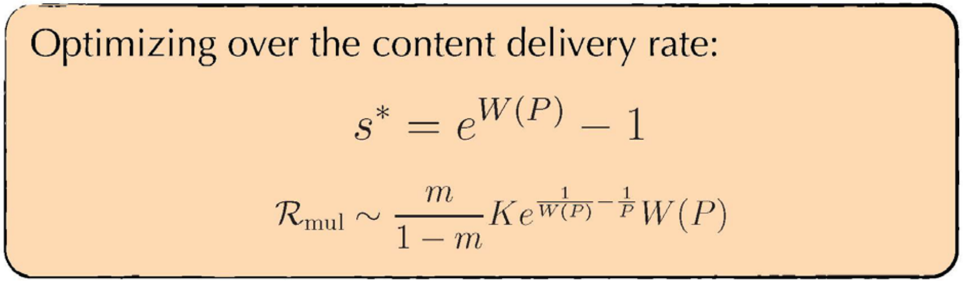
Massive MIMO



Coded caching

One bit of feedback is enough for scalability)

- Don’t send to users with bad channels (SNR threshold s^*)



Optimizing over the content delivery rate:

$$s^* = e^{W(P)} - 1$$

$$\mathcal{R}_{\text{mul}} \sim \frac{m}{1-m} K e^{\frac{1}{W(P)} - \frac{1}{P}} W(P)$$

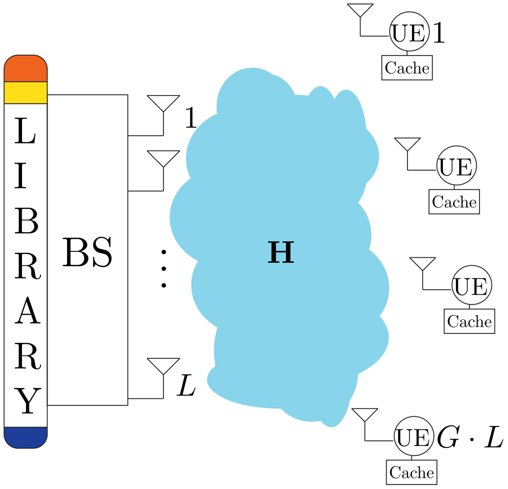
“Scalable wireless content delivery with coded caching” Ngo-Kobayashi-Yang 2017

Challenge – CSI Feedback

Challenge

Coded Caching and CSI Feedback

Coded Caching in the Overloaded MISO BC



Setting

- L antennas
- $K = G \cdot L$ Users
- $\gamma \in (0, 1)$ fractional cache

Consequences

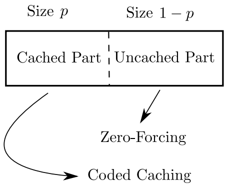
- Multiplexing needs G slots
- If $K\gamma \leq L - 1 \Rightarrow$ ZF outperforms Coded Caching

Proposed Solution

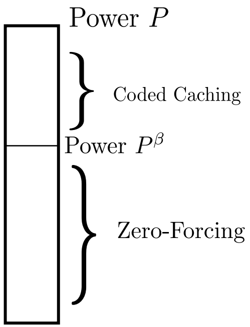
Combine the two

Coded Caching in the Overloaded MISO BC

File Allocation



Power Allocation

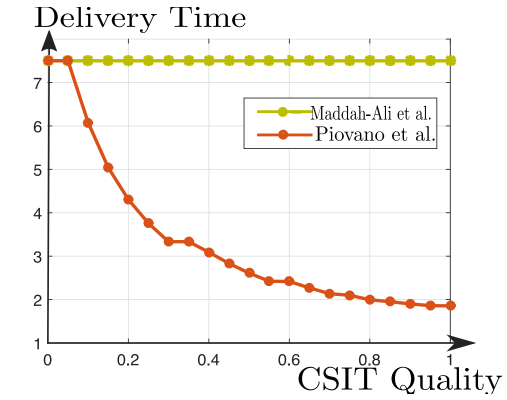


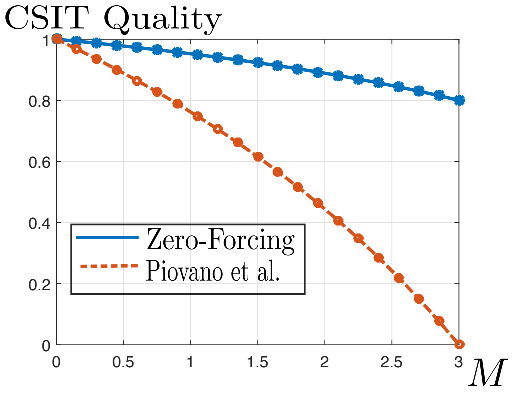
Delivery Time

$$T^\star(p) = \frac{K}{L}(1-p) + \frac{K(p-\gamma)}{\frac{K\gamma}{p}+1}$$

Goal

Optimize with respect to p





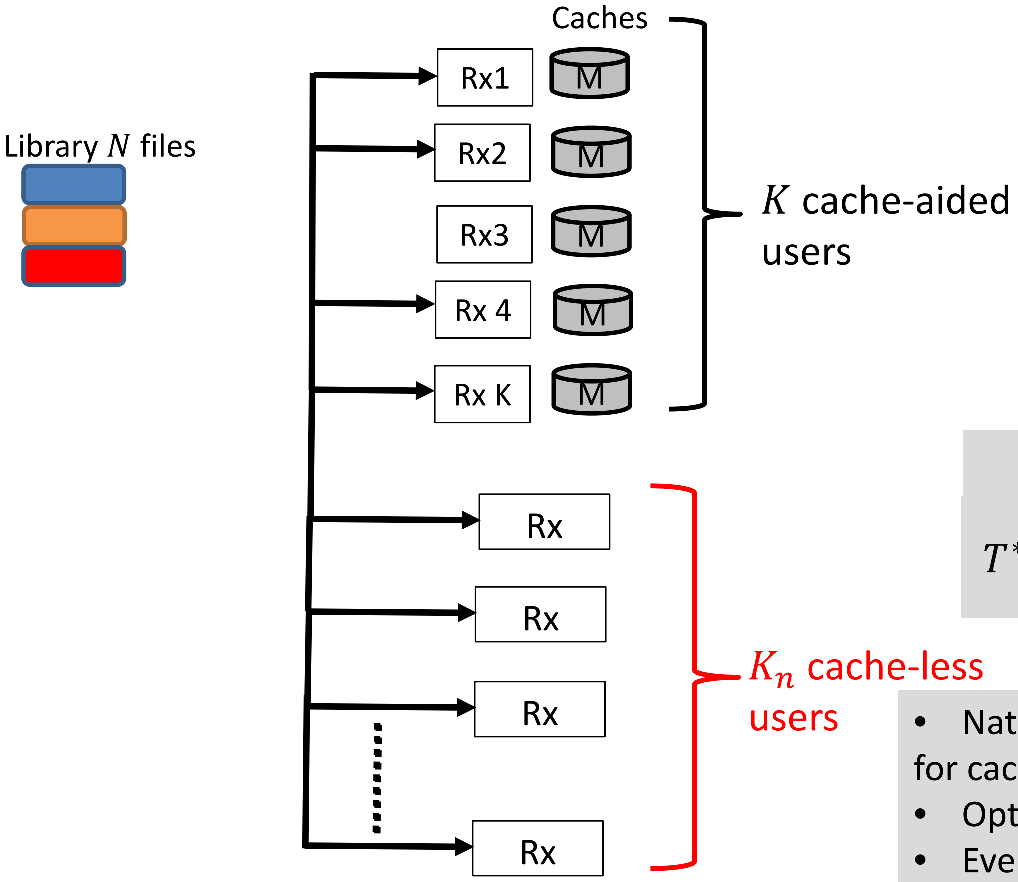
[Piovano et al '17]

Challenge – Cache asymmetry

Challenge

Cache-size Non-Homogeneity

Coded Caching with Cache-less Users



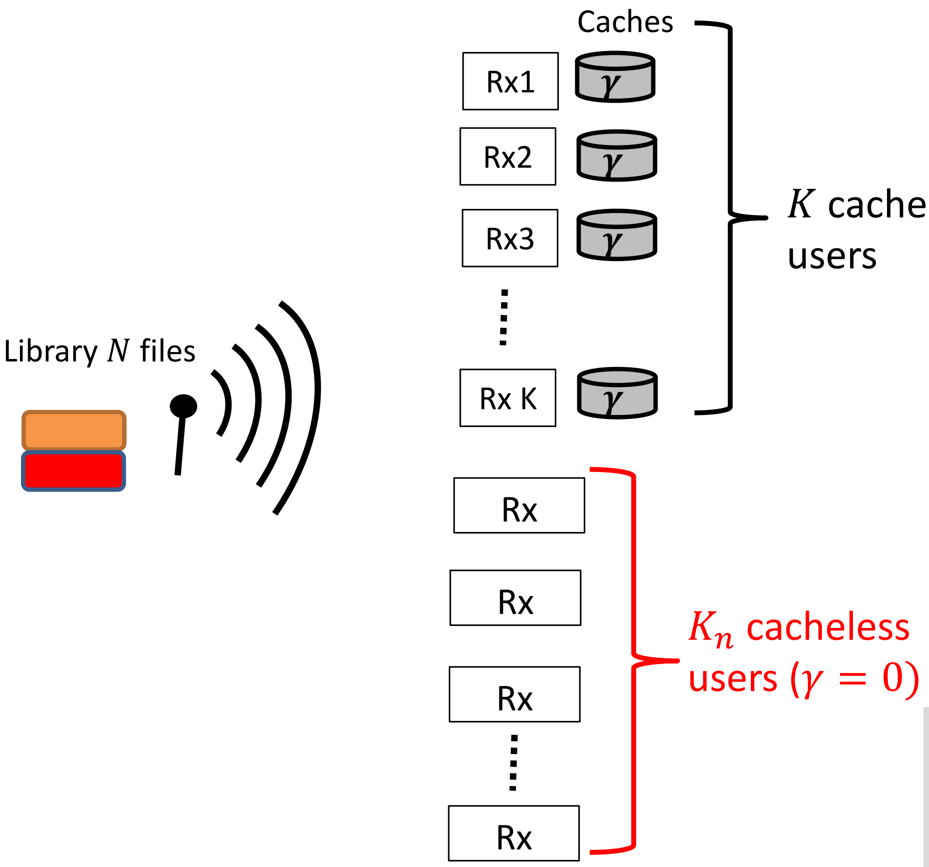
$$T^* = \frac{K(1-\gamma)}{1+K\gamma}$$

Optimal

$$T^* = \frac{K(1-\gamma)}{1+K\gamma} + K_n$$

- Naturally no caching gain for cache-less users
- Optimal: treat separately
- Every time we add a cache-less users, we pay full penalty

Same for SISO Coded Caching



$$T^* = \frac{K(1-\gamma)}{1+K\gamma}$$

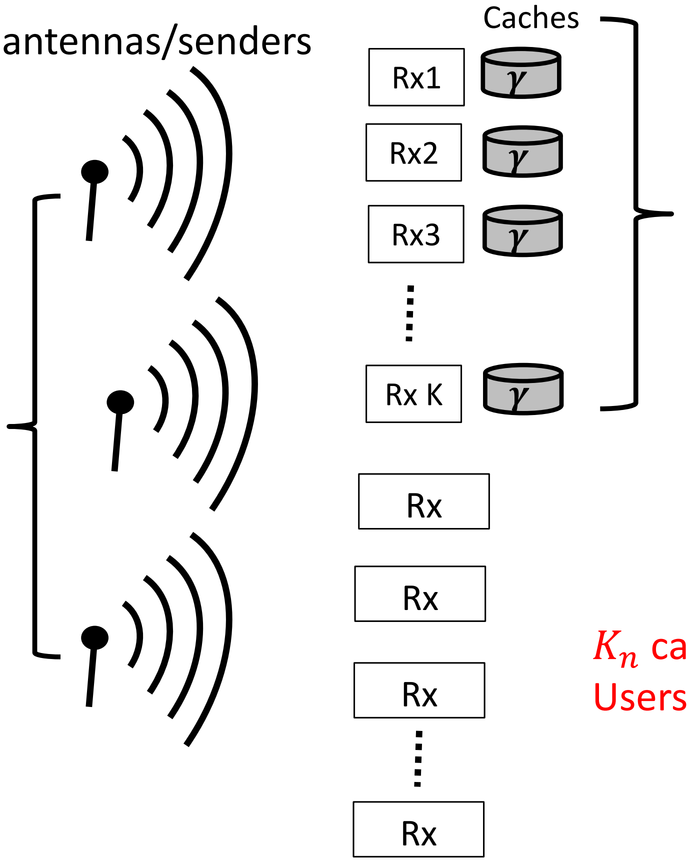
Optimal

$$T^* = \frac{K(1-\gamma)}{1+K\gamma} + K_n$$

- Naturally no caching gain for cache-less users
- Optimal: treat separately
- Every time we add a cache-less users, we pay full penalty

Separated Approach with Multiple Antennas

L antennas/senders



K cache-aided users

K_n cacheless Users $(\gamma = 0)$

$$T^* = \frac{K(1-\gamma)}{L + K\gamma}$$

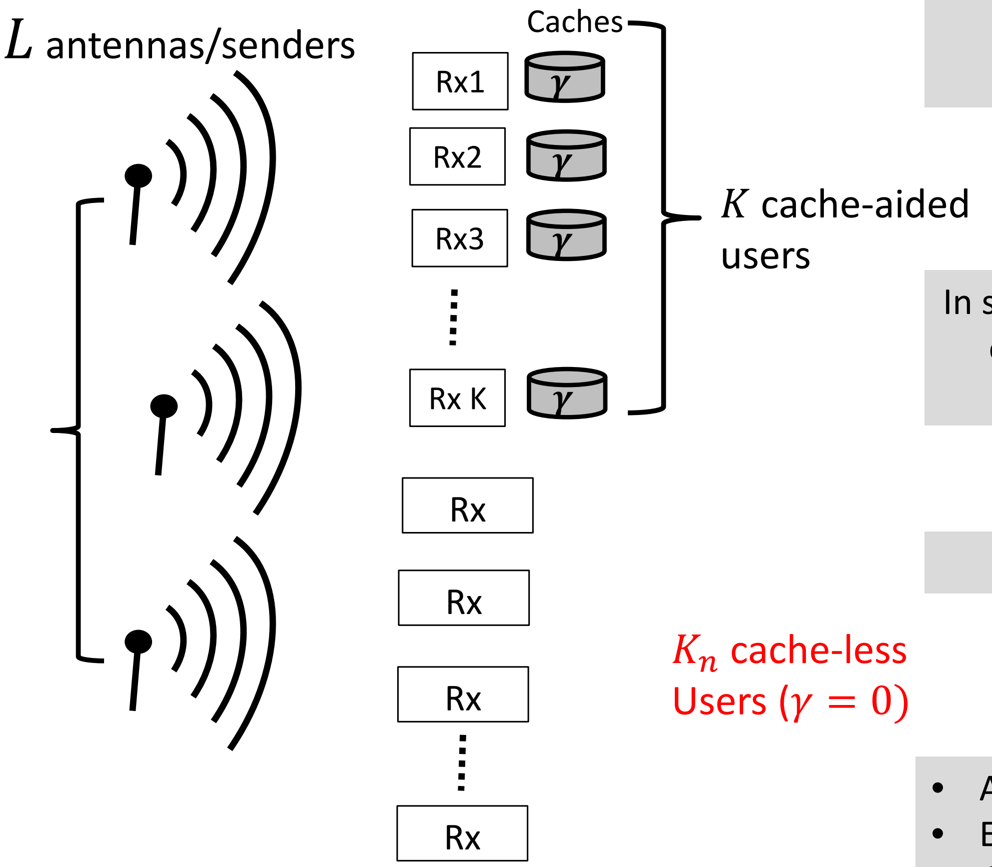
$$DoF = L + K\gamma$$

(Shariatpanahi et al. 2016)

Separated approach with multiple antennas

$$T_{sep} = \frac{K(1-\gamma)}{L+K\gamma} + \frac{K_n}{L}$$

MISO Coded Caching with Cache-less Users (separate)



Separated approach with multiple antennas

$$T_{sep} = \frac{K(1-\gamma)}{L+K\gamma} + \frac{K_n}{L}$$

In separated approach, naturally only cache-aided users get both gains $L + K\gamma$

We will show

$$T^* = \frac{K(1-\gamma)}{L+K\gamma} + \frac{K_n}{L+K\gamma}$$

- All users get full gains $L + K\gamma$
- Even cache-less users can get full caching gain $K\gamma$

Design

Basic parameters

- τ: $K\gamma$ cache-aided users
- ϕ: cache-aided user with precoding
- $\chi = \tau \cup \{\phi\}$: XOR of $K\gamma + 1$ files
- g_t: group of $L-1$ cacheless users

$$\mathbf{x}^{t}_{\tau,\phi} = \mathcal{H}^{-1}_{\phi \cup g_t} \begin{bmatrix} \bigoplus_{k \in \chi} W^{\phi_k}_{d_k, \chi \setminus \{k\}} \\ W^{\phi}_{d_{g_t(1)},\tau} \\ \vdots \\ W^{\phi}_{d_{g_t(L-1)},\tau} \end{bmatrix}$$

Understanding the source of the gains

- Realization: to get full DoF, *not all* served users require caches
- Allows substituting $L-1$ cache-aided users per transmission with cache-less users
- These substituted caches can be used in later transmissions to boost performance.

[0] Lampiris-Elia 18

Main theorem

We will use $T_1 \triangleq T_1(K,\gamma) = \frac{K(1-\gamma)}{1+K\gamma}$ $(T_1 \approx \frac{1}{\gamma})$.

Theorem

In the MISO BC with L antennas, K cache-aided users, fractional cache size γ, and $K_n \geq (L-1)T_1$ cache-less users, the delivery time

$$T = \frac{(L-1)T_1 + K(1-\gamma)}{K\gamma + L} + \frac{K_n - (L-1)T_1}{\min\{L, K_n - (L-1)T_1\}}$$

is achievable and within a factor of 2 from optimal, while if $K_n \leq (L-1)T_1(K,\gamma)$ then

$$T = \frac{K_n + K(1-\gamma)}{K\gamma + L}$$

is achievable and within a factor of 3 from optimal.

Ramifications: Increasing impact of adding antennas

Context: In completely cache-less system

In completely cache-less K_n-user MISO BC, adding one more antenna allows (without added delay costs) the addition of only a diminishing number of $\frac{K_n}{L}$ extra cache-less users.

Corollary

Starting from the single-stream BC with K cache-aided users (γ), then adding an extra $L-1$ transmit antennas, allows for the addition of

$$(L-1)T_1(K,\gamma) \approx \frac{L-1}{\gamma}$$

extra cache-less users, at no added delay costs.

Example: Full caching gain for cacheless users

Example ($\gamma \approx 1/100, K = 600, T_1 = 99$)

- Add 5 antennas ($L = 6$)
- Can add ≈ 500 cache-less users
- Served with full caching $+$ multiplexing gain
- Total delay $T = T_1 = 99$

Example: Multiplicative DoF boosts from increasing L

Example ($\gamma \approx 1/100, K = 600, T_1 = 99$)

- Observing multiplicative effect of adding antennas

Example: Multiplicative DoF boosts from increasing L

Only cache-aided users ($K_n = 0$)

- Add 1 antenna: delay reduction 1.16
- Add 2 antennas: delay reduction 1.33
- Add 3 antennas: delay reduction 1.5

Cacheless and cache-aided coexistence ($K_n = 300$) ($L' = 4$)

- Add 1 antenna: delay reduction 2
- Add 2 antennas: delay reduction 3

Example: Full caching gain for cacheless users

Traditional MISO system (no caching)

- MISO traditional (cache-less) systems
- $L = 50$ tx-antennas
- Serving $K_n = 1000$ users
- Per-user $DoF = \frac{50}{1000} = \frac{1}{20}$

MISO system - Now add cache-aided users

- Can add an UNLIMITED number of cache-aided users with $\gamma \geq 1/20$
- at delay cost of less than 2%

Ramifications: Multiple antennas for ‘balancing’ cache-size unevenness

Context: Cache-size unevenness generally hurts performance

In single-stream, having cache-size asymmetry generally causes delay penalties (best is uniform cache sizes).

Corollary

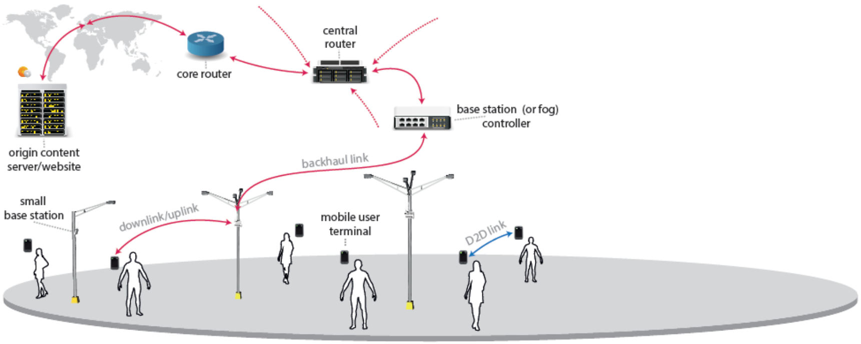
The (K, γ, K_n) *MISO BC with* $K_n \leq (L-1)\frac{K(1-\gamma)}{1+K\gamma}$ *cache-less users, incurs the same achievable delay*

$$T(K, \gamma, K_n) = \frac{K_n + K(1-\gamma)}{L + K\gamma} = \frac{(K+K_n)(1-\gamma_{\mathsf{av}})}{L + (K+K_n)\gamma_{\mathsf{av}}}$$

as the optimal homogeneous $(K+K_n)$*-user MISO BC with equal cache sizes* $\gamma_{\mathsf{av}} = \frac{K\gamma}{(K+K_n)}$ *(same cumulative cache* $K\gamma = (K+K_n)\gamma_{\mathsf{av}}$*).*

General Conclusions

Addressed Misconceptions



- Where to install memory
  - *"No need of deploying too many caches due to its costly nature"*
  - Now, much higher needs and gains though. Change of mind?

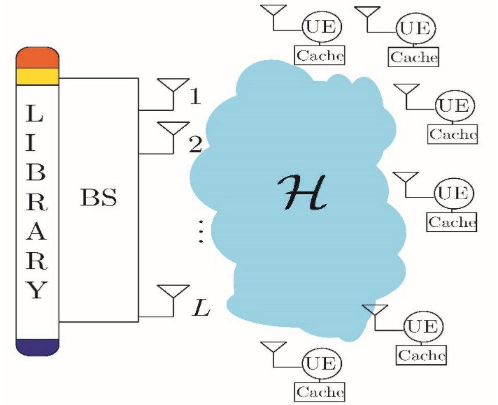
- In the future network, caches can be installed in BS, but also in mobile phones, laptops.
  - Last step can alleviate wireless capacity constraints

Source: Paschos et al. (2016)

Addressed Misconceptions

- The differences between advanced and "traditional" caching
  - Caching is an upper layer problem
  - Fusion is fascinating, and very powerful

- Example: Boosting a Multi-antenna system



Separable approach:
MISO system with 50 antennas
Users served: 50 → 55

True fusion of PHY and Coded Caching:
Same system, same constraints, same resources
Users served: 50 → 250

Addressed Misconceptions

“Coded caching can never work”

- Press Release: “Airlines can provide huge video libraries with .. Coded Caching”

- See Cadami (Coded Caching startup – Background, IT, Network coding - TUM)

- Big challenges in extending this (relatively clean) setting, to wireless

Caching in wireless: a set of different challenges

Several salient features when caching is for wireless

- Interesting tradeoffs, **synergies**, and opportunities

- Feedback and topology are unexplored frontiers in caching for wireless.
  - ➢ Among many interesting differentiating ingredients

- Certain non-separability between caching and PHY. Unbounded gains if:
  - Finite file sizes
  - Shared caches
  - Imperfect CSI

Open Problems and Future Directions

- Fusing PHY and CC to improve performance and subpacketization
  - Need to boost DoF gains for small γ
  - Under subpacketization constraints
  - Need to explore new cache-endowed powerful PHY resources
- CC in different network topologies.
  - Topologies affect FB, interference, and multicasting capabilities (all connected)

Open Problems and Future Directions

Subpacketization bottleneck

STILL A MASSIVE PROBLEM:

- Expect $S_{max} \leq 10^5$(generous) – especially for streaming, (closer to 10^4)

BAD NEWS:

- In single stream coded caching, no known algorithm can achieve gain $G > 5$
  - If $\gamma \leq \frac{1}{20}$ and $S_{max} \leq 10^5$

GOOD NEWS:

- With multi-antenna (multi-sender) decomposition, any new algorithm's improvement, will be multiplied by a factor of up to L

Open Problems and Future Directions

- Computational and implementation complexity (subpacketization, clique-finding, cache-allocation)

- Increasing effective cache size.
  - Machine Learning
  - Recommendations

Open Problems and Future Directions

Outer bounds

- Main challenges:
  - Asymmetry (Parrinello et al ITW 2018),
  - Topology
  - Feedback (Quality: Piovano et al)
  - Multiple senders/antennas (Naderializadeh et al – 2017)
- We are just beginning to scratch the surface.

Open Problems and Future Directions

Finite SNR – error prone channels

- Mostly considered error-free communications

- Interesting solutions if you consider finite SNR
  - Precoder optimization for multicasting (Tolli et al. 2018)
  - Unicast/multicast optimization (Shariatpanahi et al 2018).

Challenge: talking caching with network theorists

- Info theory and network theory have different methodologies
  - More emphasis on optimization under specific protocol assumptions

- Different measures of performance (beyond rate, capacity, delay, DoF, etc)
  - Infuse CC approach with network-theoretic considerations.

- Cost of cache placement
  - Mainly we have assumed zero-cost placement
  - Updating is also an issue (see `Online coded caching' – Pedarsani et al. 2016)

- More emphasis on backhaul
  - Mainly emphasizing on bandwidth reduction
  - Some works: Tang et al 2018, see also Sengupta et al 2016

Challenge: speaking to network theorist caching experts

- Caching with secure communications (e.g. https)
  - Public key encryption changes files differently at different receivers
  - Paschos et al. , also Engelmann-Elia.
- Synchronicity
  - See Ghasemi 2018
- What is the best way to utilize file popularity and user behavior
  - Open problem and could be key in solidifying usefulness of coded caching

Open Problems and Future Directions

Distributed Computing

- Sructured Reduce Functions: For many problems, the Reduce function may be linear. A worker node only needs a linear function of the other nodes at the end of the Map Phase.
  - Leveraging this structure[9] needs to be systematically investigated.
- Reducing Shuffle phase traffic & mitigating stragglers simultaneously: In practice, MapReduce systems also suffer from the problem of stragglers, where some workers are slower. Coding techniques to simultaneously address both issues are of interest.
- Asymmetry in job assignments: Impact of asymmetry in task assigments needs a systematic investigation. Could be useful for certain job types.

[9] S. Li, M. A. Maddah-Ali and A. S. Avestimehr, Compressed coded distributed computing, *IEEE Intl. Symp. on Information Theory, 2018.*

Open Problems and Future Directions

Distributed Computing

- Systems Challenges: Current distributed computing paradigms such as MapReduce, Spark, Hadoop etc. do not allow a user to specify the location of jobs.
    - Coded Distributed Computing requires “Subpacketize and Place (carefully)"; requires engaging the networking and distributed systems community

THANKS FOR YOUR ATTENTION!

References

Subpacketization

- *K. Shanmugam, M. Ji, A. M. Tulino, J. Llorca, and A. G. Dimakis., “Finite-length analysis of caching-aided coded multicasting,” IEEE Transactions on Information Theory, Oct. 2016.*
- *Q. Yan, M. Cheng, X. Tang, and Q. Chen, “On the placement delivery array design for centralized coded caching scheme,” IEEE Transactions on Information Theory, Sept. 2017.*
- *L. Tang and A. Ramamoorthy, “Low subpacketization schemes for coded caching,” ISIT 2017.*
- *C. Shangguan, Y. Zhang, and G. Ge, “Centralized coded caching schemes: A hypergraph theoretical approach,” IEEE Transactions on Information Theory, Aug. 2018.*

References

Subpacketization

- *C. Shangguan, Y. Zhang, and G. Ge, "Centralized coded caching schemes: A hypergraph theoretical approach," IEEE Transactions on Information Theory, Aug. 2018.*
- *K. Shanmugam, A. M. Tulino, and A. G. Dimakis, "Coded caching with linear subpacketization is possible using Ruzsa-Szemeredi graphs," ISIT 2017.*
- *S. Jin, Y. Cui, H. Liu, and G. Caire, "Order-optimal decentralized coded caching schemes with good performance in finite file size regime," GLOBECOM 2016.*
- *E. Lampiris and P. Elia, "Adding transmitters dramatically boosts coded-caching gains for finite file sizes," IEEE Journal on Selected Areas in Communications (Caching), June 2018.*
- *E. Lampiris and P. Elia, "Achieving full multiplexing and unbounded caching gains with bounded feedback resources," ISIT 2018.*

References

Multiple senders/antennas

- *S. P. Shariatpanahi, S. A. Motahari, and B. H. Khalaj, "Multi-server coded caching," IEEE Trans. Inf. Theory, Dec. 2016.*
- *N. Naderializadeh, M. A. Maddah-Ali, and A. S. Avestimehr, "Fundamental limits of cache-aided interference management," IEEE Trans. Inf. Theory, May 2017.*
- *A. Sengupta, R. Tandon, and O. Simeone, "Cache aided wireless networks: Tradeoffs between storage and latency," arXiv:1512.07856, 2015.*
- *Y. Cao, M. Tao, F. Xu, and K. Liu, "Fundamental storage-latency tradeoff in cache-aided MIMO interference networks," arXiv:1609.01826, 2016.*
- *J. Hachem, U. Niesen, and S. N. Diggavi, "Degrees of Freedom of cache-aided wireless interference networks," arXiv:1606.03175, 2016.*
- *J. S. P. Roig, F. Tosato, and D. Gündüz, "Interference networks with caches at both ends," arXiv:1703.04349, 2017.*
- *S. Yang, K. H. Ngo, and M. Kobayashi, "Content delivery with coded caching and massive MIMO in 5G," ISTC 2016.*

References

Multiple senders/antennas

- *S. P. Shariatpanahi, G. Caire, and B. H. Khalaj, "Multi-antenna coded caching," arXiv:1701.02979, 2017.*
- *E. Piovano, H. Joudeh, and B. Clerckx, "On coded caching in the overloaded MISO broadcast channel," arXiv:1702.01672, 2017.*
- *J. Zhang, F. Engelmann, and P. Elia, "Coded caching for reducing CSIT-feedback in wireless communications," Allerton Conf. Communication, Control and Computing, 2015.*
- *J. Zhang and P. Elia, "Feedback-aided coded caching for the MISO BC with small caches," arXiv:1606.05396, 2016.*
- *M. A. Maddah-Ali and U. Niesen, "Cache-aided interference channels," ISIT 2015.*
- *Y. Cao and M. Tao, "Treating content delivery in multi-antenna coded caching as general message sets transmission: A DoF region perspective," arxiv: 1807.01432, 2018.*
- *E. Piovano, H. Joudeh, and B. Clerckx, "Robust cache-aided interference management under full transmitter cooperation," ISIT 2018.*

References

Scalability in networks

- *M. Bayat, R. K. Mungara, and G. Caire, "Achieving spatial scalability for coded caching over wireless networks," arXiv:1803.05702, 2018*

- *K.-H. Ngo, S. Yang, and M. Kobayashi, "Scalable content delivery with coded caching in multi-antenna fading channels," IEEE Transactions on Wireless Communications, 2018.*

References

Topology

- *X. Yi and G. Caire, "Topological coded caching," ISIT 2016.*
- *J. Zhang and P. Elia, "Wireless Coded Caching: A topological perspective," ISIT 2017.*
- *A. Destounis, M. Kobayashi, G. Paschos, and A. Ghorbel, "Alpha fair Coded Caching," WiOpt 2017.*
- *S. S. Bidokhti, M. Wigger, and R. Timo, "Erasure broadcast networks with receiver caching," ISIT 2016.*
- *E. Lampiris, J. Zhang, and P. Elia, "Cache-aided cooperation with no CSIT," ISIT 2017.*
- *E. Piovano, H. Joudeh, and B. Clerckx, "Generalized degrees of freedom of the symmetric cache-aided MISO broadcast channel with partial CSIT," arXiv:1712.05244, 2017.*

References

Heterogeneous cache sizes

- *S. S. Bidokhti, M. Wigger, and R. Timo, “Erasure broadcast networks with receiver caching,” ISIT 2016.*
- *S. Wang, W. Li, X. Tian, and H. Liu, “Fundamental limits of heterogenous cache,” arxiv: 1504.01123, 2015.*
- *M. M. Amiri, Q. Yang, and D. Gündüz, “Decentralized caching and coded delivery with distinct cache capacities,” IEEE Transactions on Communications, 2017.*
- *A. Sengupta, R. Tandon, and T. C. Clanc, “Layered caching for heterogeneous storage,” Asilomar Conference on Signals, Systems and Computers, 2016.*
- *A. M. Ibrahim, A. A. Zewail, and A. Yener, “Centralized coded caching with heterogeneous cache sizes,” WCNC 2017.*
- *Eleftherios Lampiris and Petros Elia, "Full Coded Caching Gains for Cache-less Users," ITW 2018.*

References

Multiple demands, Shared caches

- *H. Xu, C. Gong, and X. Wang, "Efficient file delivery for coded prefetching in shared cache networks with multiple requests per user," arxiv 1803.09408, 2018.*
- *M. Ji, A. M. Tulino, J. Llorca, and G. Caire, "Caching-aided coded multicasting with multiple random requests," ITW 2015.*
- *A. Sengupta and R. Tandon, "Improved approximation of storage-rate tradeoff for caching with multiple demands," IEEE Transactions on Communications, May 2017.*
- *L. Zhang, Z. Wang, M. Xiao, G. Wu, and S. Li, "Centralized caching in two-layer networks: Algorithms and limits," WiMob 2016.*
- *N. Karamchandani, U. Niesen, M. A. Maddah-Ali, and S. N. Diggavi, "Hierarchical coded caching," IEEE Transactions on Information Theory, June 2016.*
- *Y. Wei and S. Ulukus, "Coded caching with multiple file requests," Allerton Conference on Communication, Control, and Computing, 2017.*
- *E. Parrinello, A. Ünsal, and P. Elia, "Fundamental limits of coded caching with shared caches and uncoded prefetching," arxiv: 1809.09422, 2018.*

References

Index coding in coded caching and distributed computing

- *M. Li, L. Ong, and S. J. Johnson, "Cooperative multi-sender index coding," arxiv: 1701.03877v4, 2017.*
- *P. Sadeghi, F. Arbabjolfaei, and Y. H. Kim, "Distributed index coding," ITW 2016.*

References

Distributed computing and data shuffling

- *S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, "A fundamental tradeoff between computation and communication in distributed computing," IEEE Transactions on Information Theory, Jan 2018.*
- *E. Parrinello, E. Lampiris, and P. Elia, "Coded distributed computing with node cooperation substantially increases speedup factors," ISIT 2018.*
- *K. Konstantinidis and A. Ramamoorthy, "Leveraging coding techniques for speeding up distributed computing,", arxiv: 1802.03049, 2018.*
- *Q. Yan, S. Yang, and M. A. Wigger, "A storage-computation-communication tradeoff for distributed computing," arxiv: 1805.10462, 2018.*
- *N. Woolsey, R. Chen, and M. Ji, "A new combinatorial design of coded distributed computing," ISIT 2018.*
- *M. A. Attia and R. Tandon, "Information theoretic limits of data shuffling for distributed learning," GLOBECOM 2016.*
- *M. A. Attia and R. Tandon, "Approximately optimal distributed data shuffling," ISIT 2018.*
- *K. Wan, D. Tuninetti, M. Ji, and P. Piantanida, "Fundamental limits of distributed data shuffling," arxiv: 1807.00056, 2018.*
- *A. Elmahdy and S. Mohajer, "On the fundamental limits of coded data shuffling," ISIT 2018.*

References

Finite SNR

- *A. Tölli, S. P. Shariatpanahi, J. Kaleva, and B. Khalaj, "Multiantenna interference management for coded caching," arXiv:1711.03364, 2017*
- *Seyed Pooya Shariatpanahi, Giuseppe Caire, Babak Hossein Khalaj, "Physical-Layer Schemes for Wireless Coded Caching" arXiv:1711.05969, 2017*

References

Limitations of current communication paradigms：

- *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update 2014–2019. White Paper.*
- *J. Andrews, X. Zhang, G. Durgin, A. Gupta, "Are we approaching the fundamental limits of wireless network densification?," ArXiv e-prints, Dec. 2015.*
- *L. Zheng and D. N. C. Tse, "Communication on the Grassmann manifold: A geometric approach to the noncoherent multiple-antenna channel," IEEE Trans. Information Theory, Feb. 2002.*
- *A. Lozano, R. W. Heath Jr., and J. G. Andrews, "Fundamental limits of cooperation," IEEE Trans. Information Theory, Sep. 2013.*
- *M. A. Maddah-Ali and D. N. C. Tse, "Completely stale transmitter channel state information is still very useful," IEEE Trans. Information Theory, Jul. 2012.*
- *A. Singh, P. Elia, J. Jaldén, "Achieving a vanishing SNR-gap to exact lattice decoding at a subexponential complexity," IEEE Trans. Information Theory, Jun. 2012.*
- *A. J. Fehske, et al., "The global footprint of mobile communications: The ecological and economic perspective," IEEE Communications Magazine, Aug. 2011.*
- *T. M. Cover and J. A. Thomas, Elements of information theory. John Wiley & Sons, 2012.*

References

Feedback communication theoretic solutions：

- V.R. Cadambe, S.A. Jafar, “Interference alignment and degrees of freedom of the K-User interference channel,” IEEE Trans. Information Theory, Aug. 2008.
- J. Chen and P. Elia, “Toward the performance versus feedback tradeoff for the two-user MISO broadcast channel," IEEE Trans. Information Theory, Dec. 2013.
- J. Chen, P. Elia, and S. Jafar, “On the two-user MISO broadcast channel with alternating CSIT: A topological perspective,” IEEE Trans. Information Theory, Aug. 2015.
- R. Tandon, S. A. Jafar, S. Shamai, and H. V. Poor, “On the synergistic benefits of alternating CSIT for the MISO broadcast channel,” IEEE Trans. Information Theory, Jul. 2013.
- C. S. Vaze, S. Karmakar, and M. K. Varanasi, “On the generalized degrees of freedom region of the MIMO interference channel with no CSIT,” IEEE International Symposium on Information Theory (ISIT), Aug. 2011.
- J. Chen, S. Yang, A. Ozgur, A. Goldsmith, “Achieving Full DoF in Heterogeneous Parallel Broadcast Channels with Outdated CSIT”, IEEE Trans. Information Theory,

  Sep. 2014
- P. de Kerret, D. Gesbert, J. Zhang and P. Elia, “Optimal sum-DoF of the K-user MISO BC with current and delayed feedback,” ArXiv e-prints, Apr. 2016

References

Earlier works: Single-stream Coded caching

- M. Maddah-Ali and U. Niesen, “Fundamental limits of caching,” IEEE Trans. Information Theory, May 2014.
- U. Niesen and M. Maddah-Ali, “Coded caching with non-uniform demands,” in Computer Communications Workshops (INFOCOM WKSHPS), May 2014.
- M. Maddah-Ali and U. Niesen, “Decentralized caching attains order-optimal memory-rate tradeoff,” Allerton Conference, Monticello, IL, Oct. 2013 – see also ArXiv e-prints, 2013.
- K.Wan, D. Tuninetti, and P. Piantanida, “On Caching with More Users than Files”, ArXiv e-prints, Jan. 2016.
- K. Shanmugam, M. Ji, A. M.Tulino, J. Llorca, and A. G. Dimakis, “Finite length analysis of caching-aided coded multicasting,” ArXiv e-prints, Aug. 2015.
- Z. Chen, P. Fan, and K.B. Letaief, “Fundamental Limits of Caching: Improved Bounds For Small Buffer Users”, ArXiv e-prints, Nov. 2015.
- M. Ji, A. M. Tulino, J. Llorca, G. Caire, “Order-Optimal Rate of Caching and Coded Multicasting with Random Demands”, ArXiv e-prints, Feb. 2015.
- S. Sahraei, M. Gastpar, “Multi-Library Coded Caching”, ArXiv e-prints, Jan. 2016

References

Earlier works: Single-stream Coded caching

- M. Maddah-Ali and U. Niesen, “Fundamental limits of caching,” IEEE Trans. Information Theory, May 2014.
- M. Mohammadi Amiri, D. Gunduz, “Fundamental Limits of Caching: Improved Delivery Rate-Cache Capacity Trade-off”, ArXiv e-prints, Apr. 2016.
- R. Pedarsani, M. Ali Maddah-Ali, U. Niesen, “Online Coded Caching”, ArXiv e-prints, Nov. 2013.
- J. Hachem, N. Karamchandani, S. Diggavi, “Effect of Number of Users in Multi-Level Coded Caching”, ArXiv e-prints, Apr. 2015
- J. Zhang, X. Lin, X. Wang, “Coded Caching under Arbitrary Popularity Distributions”, Information Theory and Applications Workshop (ITA), Feb. 2015
- C. Wang, S. Lim, M. Gastpar, “Information-Theoretic Caching: Sequential Coding for Computing”, ArXiv e-prints, Feb. 2016
- U. Niesen, M. Maddah-Ali, “Coded Caching for Delay-Sensitive Content”, ArXiv e-prints, Jul. 2014.
- S. Bidokhti, M. Wigger, R. Timo, “Noisy Broadcast Networks with Receiver Caching”, ArXiv e-prints, May. 2016.

References

Extensions of Coded Caching in different settings:

- R. Timo and M. A. Wigger, “Joint cache-channel coding over erasure broadcast channels,” ArXiv e-prints, May 2015.
- A. Ghorbel, M. Kobayashi, S. Yang, “Cache-Enabled Broadcast Packet Erasure Channels with State Feedback”, Allerton Conference, Monticello, IL, Oct. 2015.
- N. Karamchandani, U. Niesen, M. Maddah-Ali, S. Diggavi, “Hierarchical Coded Caching”, ArXiv e-prints, Jun 2014.
- K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, G. Caire"FemtoCaching: Wireless video content delivery through distributed caching helpers", ArXiv e-prints, Sep 2013
- M. Ji, G. Caire, A. F. Molisch, “Fundamental limits of distributed caching in D2D wireless networks”, ArXiv e-prints, Apr 2013
- Y. Ugur, Z. Awan, A. Sezgin, "Cloud Radio Access Networks With Coded Caching", ArXiv e-prints Feb 2016
- S. Lim, C. Wang, M. Gastpar, “Information Theoretic Caching: The Multi-User Case”, ArXiv e-prints Apr 2016

References

- V. Bioglio, F. Gabry, I. Land, "Optimizing MDS Codes for Caching at the Edge", ArXiv e-prints Sep 2015.
- Altman, E., Avrachenkov, K. and Goseling, J. “Distributed storage in the plane”. In Networking Conference, 2014 IFIP (pp. 1-9). IEEE.
- Avrachenkov, K., Bai, X. and Goseling, J., 2016. “Optimization of Caching Devices with Geometric Constraints,” arXiv preprint arXiv:1602.03635.
- B. Perabathini, E. Baştuğ, M. Kountouris, M. Debbah, A. Conte, “Caching at the Edge: a Green Perspective for 5G Networks”, ArXiv e-prints Mar 2015.
- M. Deghel, E. Bastug, M. Assaad, and M. Debbah, “On the benefits of edge caching for MIMO interference alignment,” in Signal Processing Advances in Wireless Communications (SPAWC), Jun. 2015.
- A. Liu, V.t K. N. Lau, “Cache-Enabled Opportunistic Cooperative MIMO for Video Streaming in Wireless Systems”, IEEE Trans. Signal Processing, Jan. 2014.
- S.-E. Elayoubi and J. Roberts, “Performance and cost effectiveness of caching in mobile access networks,” in Proc. of the 2nd International Conference on Information-Centric Networking, 2015.

References

Coded Caching with Feedback

- *E. Lampiris and P. Elia, "Achieving full multiplexing and unbounded caching gains with bounded feedback resources," ISIT 2018.*
- *E. Lampiris, J. Zhang, and P. Elia, "Cache-aided cooperation with no CSIT," ISIT 2017.*
- *E. Piovano, H. Joudeh, and B. Clerckx, "On Coded Caching in the overloaded MISO Broadcast Channel," ISIT 2017.*
- *E. Piovano, H. Joudeh, and B. Clerckx, "Generalized degrees of freedom of the symmetric cache-aided MISO broadcast channel with partial CSIT," arXiv:1712.05244, 2017.*
- *J. Zhang and P. Elia, "Fundamental limits of cache-aided wireless BC: Interplay of coded-caching and CSIT feedback," IEEE Transactions on Information Theory, May 2017*
- *K.-H. Ngo, S. Yang, and M. Kobayashi, "Scalable content delivery with coded caching in multi-antenna fading channels," IEEE Transactions on Wireless Communications, 2018.*
- *A. Liu, V. K. N. Lau, "Mixed-Timescale Precoding and Cache Control in Cached MIMO Interference Network", IEEE Trans. Signal Processing, Dec. 2013.*
- *J. Zhang, F. Engelmann and P. Elia, "Coded caching for reducing CSIT-feedback in wireless communications," Allerton Conference, Oct. 2015.*

References

Newer outer bounds

- *K. Wan, D. Tuninetti, and P. Piantanida, “On the optimality of uncoded cache placement,” ITW 2016.*
- *Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, “The exact rate-memory tradeoff for caching with uncoded prefetching,” IEEE Transactions on Information Theory, Feb. 2018.*
- *Emanuele Parrinello, Ayse Ünsal and Petros Elia, "Fundamental Limits of Caching in Heterogeneous Networks with Uncoded Prefetching", Submitted Trans IT, Oct. 2018*
- H. Ghasemi and A. Ramamoorthy, “Improved lower bounds for coded caching,” *IEEE Transactions on Information Theory, July 2017.*
- S. Lim, C. Wang, M. Gastpar, “Information Theoretic Caching: The Multi-User Case”, *IEEE Transactions on Information Theory, Nov. 2017.*
- Q. Yan, X. Tang, Q. Chen, "On the Gap Between Decentralized and Centralized Coded Caching Schemes," arXiv:1605.04626v1, 2016.
- *C. Wang, S. H. Lim, and M. Gastpar, “Information-theoretic caching: Sequential coding for computing,” IEEE Transactions on Information Theory, Nov. 2016*

References

Non uniform demands

- *U. Niesen and M. A. Maddah-Ali, "Coded caching with nonuniform demands," IEEE Transactions on Information Theory, Feb. 2017.*

- *J. Zhang, X. Lin, and X. Wang, "Coded caching under arbitrary popularity distributions," IEEE Transactions on Information Theory, Jan. 2018.*

- *M. Ji, A. M. Tulino, J. Llorca, and G. Caire, "On the average performance of caching and coded multicasting with random demands," ISWCS 2014.*

References

Security

- *F. Engelmann and P. Elia, "A content-delivery protocol, exploiting the privacy benefits of coded caching," WiOpt 2017.*

- *J. Leguay, G. S. Paschos, E. Quaglia, and B. Smyth, "Cryptocache: Network caching with confidentiality," ICC 2017.*

- *A. Sengupta, R. Tandon, and T. C. Clancy, "Fundamental limits of caching with secure delivery," ICC 2014.*

- *V. Ravindrakumar, P. Panda, N. Karamchandani, and V. Prabhakaran, "Fundamental limits of secretive coded caching," ISIT 2016.*