# 2014 MoST Undergraduate Project

# Cache&Code: Peak-Time Traffic Reduction for Multimedia-on-Demand Broadcast Systems


By: Chin-Chia Hsu

Department: Electrical Engineering

Student ID. number: B00901161

Instructor: I-Hsiang Wang

March 30, 2015

**Abstract**

In this project, I aim to resolve an issue that everyone encounters in multimedia-on-demand applications: congestion, resulting from simultaneous requests during peak-traffic time, discontinues the streaming and hence users are irritated by such a nuisance.

The role of the central server can be negative or positive. For a system with a server passively fulfilling users' requests, which is indeed a branch of Index Coding problems, I start with two-receiver case and extend to several symmetric ones such as $N$-complete graph, $N$-directed cycle and $N$-receiver loop. I prove a useful theorem that facilitates the analysis on outer bounds. Additionally, I analyze the 3-directed cycle with different message lengths and verify the capacity region constructed based on this theorem.

With introduction of a placement phase and local caches, the server can actively design the caching scheme before receiving requests to reduce the load in the delivery phase. I extend the cut-set lower bound in [6] to two groups with different cache memory size, and even to multiple $G$ groups. I prove that using decentralized algorithms cannot result in universal margins over the amount of users, cache memory and total files. However, centralized algorithms performed within each group can lead to margins bounded by a constant which disappointingly grows with the number of groups. It reveals that with a more controlled caching scheme, we can still improve the performance even if we let go of the coding opportunities between groups.

Before delving into the real and complicated scenarios where each user have different non-uniform demands, I first analyze the basic two user case with symmetric demands by proposing two intuitive caching schemes, comparing their efficiency. Then I propose a general caching scheme and spot an optimal expected rate by utilizing MATLAB and deriving the theoretical reason. The result shows that the cache memory allocation is discretized due to the **max** operations in coding. This phenomenon occurs in three user case too. It suggests that we can group files whose access probabilities fall in the same range and assign the equal size of cache memory to each file within the same group but different among groups depending on the relative probability range. In other words, we may reach optimal rates by transforming the non-uniform demands into discretized ones.

# Content


**1 Introduction** **2**

1.1 Literature Review and Discussion . . . 2

1.1.1 Index Coding . . . 2

1.1.2 Subscription Caching System . . . 3

1.2 Problem Description . . . 3

1.2.1 Index Coding . . . 3

1.2.2 Subscription Caching System . . . 4

**2 Index Coding** **5**

2.1 Illustration on Outer Bounds in 2-receiver Case . . . 5

2.2 A Useful Outer Bound . . . 6

2.2.1 $N$-directed Cycle . . . 7

2.2.2 A Special Case and Symmetric $N$-receiver Loop . . . 8

2.3 General 3-directed Cycle . . . 10

**3 Heterogeneous Caching Memory Allocation** **11**

3.1 Cut-set Bound . . . 11

3.2 Achievable Rate by Decentralized Caching Scheme . . . 12

3.3 Margin between Bounds in Decentralized Scheme . . . 13

3.4 Margin between Bounds in Centralized Scheme . . . 15

**4 Heterogeneous Preference Distribution** **17**

4.1 Proportional to Preference Caching . . . 17

4.2 Complete Files Caching . . . 19

4.3 Comparison Between Two Caching Methods . . . 19

4.4 General Caching . . . 20

4.4.1 Find Minimum $E[R]$ in General Caching in 2-user Case . . . 21

4.5 General Caching in Three-user Case . . . 22

4.5.1 Find Minimum $E[R]$ in General Caching in 3-user Case . . . 24

**5 Conclusion** **25**

# Chapter 1

# Introduction

It is unpleasant to be interrupted by a sudden pause of media player when we are watching on-line multimedia, just wondering what is happening next. It might take place too many times especially at nights, the peak-traffic time during which demands from many users paralyze the transmission. I am wondering if any methods allow users to enjoy the journey smoothly. This arising question inspires me to explore.

The high streaming during peak-traffic time leads to congestion over the network. As Internet becomes widespread, this nuisance happens frequently since watching on-line videos plays a role in our daily life: we subscribe to multimedia-on-demand service, or surf on video-sharing platforms like Youtube for visual and auditory experiences.

In the case of subscriptions to specific platforms, the central server sends bits to fulfill the users' requests. Given that users may have stored some files in their local cache before, the server can broadcast coding bits rather than the whole files to simultaneously satisfy the users' demand. It is actually one of the Index Coding problem: the sender broadcasts an encoded message based on side information at receivers such that all the receivers are able to recover their own wanted messages.

What if the contents in the local caches can be designed by the server? The server arranges which user caches which bit in advance and then lower the required bits to fulfill the users' need. Such a server actively coordinates the network. During off-peak times, the server "pushes" files in its database to the caches of users. Users quickly access the requested files if they are already in the cache memories.

If the demanded file is not stored in advance, the user "pulls" it from server, that is, the server has to stream this file over the network. The server can reduce the streaming bits by taking advantage of coding and the files already in cache memories as side information. The proposed scheme alleviates the congestion during peak-traffic time but requires careful designs both in what to cache and how to code.

Next we will review the related works on Index Coding and coded caching. I then describe the typical Index Coding problem, and extend to our subscription caching system with inserting another phase when the central server dispatches files to users' local caches.

## 1.1 Literature Review and Discussion

### 1.1.1 Index Coding

As Li, Yeung and Cai [1] broke new ground on linear coding over network to improve the transmission rate, this realm is prospering with lots of researches on diverse fields. In exploring the problems, I utilize linear coding only to achieve the outer bounds.

Z. Bar-Yossef et al. [3] used graphs to indicate the relationship of side information among receivers. They identified a measure on graphs, the minrank, and showed that the minrank is the optimal length of arbitrary Index codes for classes of side information graphs. F. Arbabjolfaei et al. [4] also represented the side information relationship in the same way. They applied outer bounds and proposed several coding schemes for analyzing the system capacity region. In the flooding discussions, I use the same graph representation in our analysis.

### 1.1.2 Subscription Caching System

In [6], Maddah-Ali and Niesen proposed a novel concept about **global caching gain** that derives from jointly optimizing both the placement and delivery phases. This new concept differs from other discussions on so-called **local caching gain** that only benefits the system if the desired file is right in the cache. They showed that the global caching gain is relevant if the aggregate global cache size $KM$ is on the order of the total amount of content $N$. They also provided theorems composed of these two caching gains and illustrated the ideas by many examples. This paper serves as a stepping-stone to the caching problem.

Popular videos attract people's attention and it's no wonder that many users may simultaneously demand those videos and then congestion happens. The popularity distribution of the files indeed have impact on the requests from users, and caching the most popular files seems optimal for a single user. However, Niesen and Maddah-Ali [7] showed that this didn't apply to the cases of multiple users. Caching only the most popular files can be highly suboptimal.

## 1.2 Problem Description

### 1.2.1 Index Coding

The Index Coding problem is depicted in Figure 1.1 . I review the introduction in [4] here. The source aims to communicate $N$ messages $W_j \in [1 : 2^{nR_j}]$, $j \in [1 : N]$, to their corresponding receivers over a broadcast, noiseless channel. The transmitted message $X^n$, perhaps after encoding, is of $n$ bits long. Each receiver $j \in [1 : N]$ has prior knowledge of $W_{\mathcal{A}_j}$, the messages of the subset $\mathcal{A}_j \subseteq [1 : N] \setminus \{j\}$. Receiver $j$ estimates $\hat{W}_j$ based on the side information $W_{\mathcal{A}_j}$ and the received symbols $X^n$.

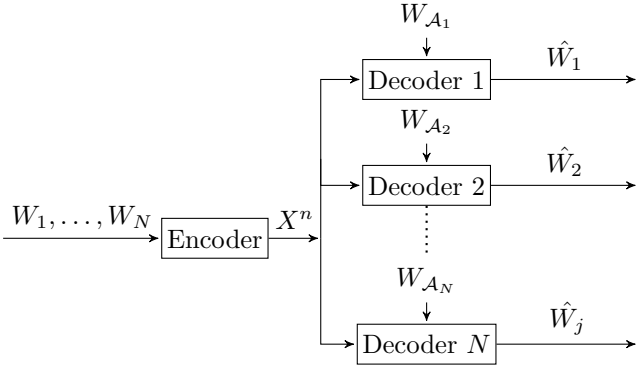



Figure 1.1: The index coding problem

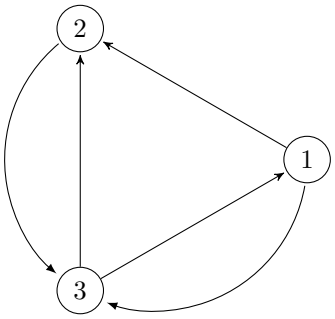



Figure 1.2: An example of directed graph

The average error probability $P_e^{(n)} = \mathbf{P}\{(\hat{W}_1, \ldots, \hat{W}_N) \neq (W_1, \ldots, W_N)\}$. A rate tuple $(R_1, \ldots, R_N)$ is achievable if there exists a sequence of $(2^{nR_1}, \ldots, 2^{nR_N}, n)$ such that $\lim_{n\to\infty} P_e^{(n)} = 0$. The closure of the set of the achievable rate tuples is the capacity region $\mathscr{C}$.

To have a clear look, I represent the relation of side informations among these receivers as a directed graph. For example, consider the 3-receiver index coding problem where $\mathcal{A}_1 = \{3\}$, $\mathcal{A}_2 = \{1,3\}$, $\mathcal{A}_3 = \{1,2\}$. Figure 1.2 is the corresponding directed graph.

The nodes represent indices of the the receivers and the edges represent the availability of side information (e.g. $2 \to 3$ means that receiver 3 has side information $W_2$). An index coding problem can be represented by a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = [1 : N]$ and $(j, k) \in \mathcal{E}$ iff $j \in \mathcal{A}_k$.

Before broadcasting, the directed graph is known to the source encoder and all receivers. Each receiver understands the decoding algorithms of all the others, and hence, can perform the same decoding process. The security or non-ideal issues are excluded—no noise, no erasure, and no eavesdropping. In the following discussions, I focus on the achievable rate tuple or sum rates, with the transmitted bits $n$ approaching infinity. I consider the achievable rates and in turn the estimation symbols are ignored.

I use linear codes to achieve the outer bounds. The topic of nonlinear codes is beyond discussions. Besides, in the symmetric cases I assume that $R_j$'s are equal, i.e. the lengths of $W_j$'s are the same.

### 1.2.2 Subscription Caching System

One single server, who has database of $N$ files $W_1,\ldots,W_N$ with $F$ bits each, transmits data to $K$ users over a broadcast and noiseless network. Each user $k$ has an isolated cache memory $Z_k$ that can store $MF$ bits for some real number $M$ between 0 and $N$.

I divide the operation of the system into two phases: a placement phase and a delivery phase. In the placement phase, the user $k$ has access to the database and able to store some of the files in the content of $Z_k$ using this database. In the delivery phase, the server becomes the only one that can access the database and knows which file the respective user requests. It transmits a message $X$ of $RF$ bits for some fixed real number $R$. The user $k$ wants to recover its requested file by using the content of $Z_k$ and received message $X$. I aim to find the minimum $R$ in different settings of the users $K$, the number of files $N$, and the number of files filled in cache $M$.

# Chapter 2

# Index Coding

The system where the passive server only acts to users' requests resembles the **Index Coding** problem. We restate the Index Coding problem and analyze some special cases by Information Theory. I try to derive the results in a more comprehensible manner, also give a proof based on simple knowledge of graph to an important corollary.

## 2.1 Illustration on Outer Bounds in 2-receiver Case

Now with the background and notations of the Index Coding problem, let's start from the simplest case where only 2 receivers exist. Based on this illustration, we can derive the outer bounds on the following cases without using the complicated techniques but still catch the ideas of Information Theory.

1. $W_{\mathcal{A}_1} = W_2$ and $W_{\mathcal{A}_2} = W_1$: Both receivers have the side information of each other as shown in Figure 2.1.

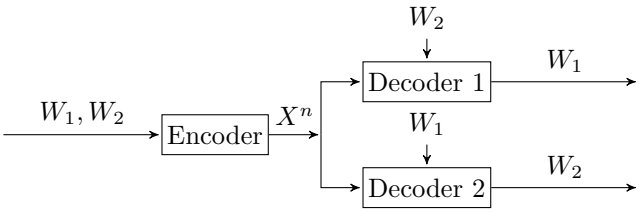



Figure 2.1: Ecoding scheme of $W_{\mathcal{A}_1} = W_2$ and $W_{\mathcal{A}_2} = W_1$

   By Information Theory, the inferred informations, independent statistically with the side informations, cannot be larger than the information in $X^n$. The bits of $W_1$—an alternative measure of information—cannot be over that of $X_n$. That is, $nR_1 \leq n, \therefore R_1 \leq 1$. Similarly, $R_2 \leq 1$. $R_1 = R_2 = 1$ when the encoder **xor**s $W_1$ and $W_2$, and both of the decoders **xor**s $X_n$ and respective side information to recover their own messages. The capacity region is thus the square with vertices (0,0), (0,1), (1,0) and (1,1).

2. $W_{\mathcal{A}_1} = \emptyset$ and $W_{\mathcal{A}_2} = W_1$: One of the receivers (here assume receiver 1) has the side information of the other, while the other does not. It is depicted in Figure 2.2.

   For receiver 2, the discussion is like the previous one. For receiver 1, however, he can not only recover $W_1$ but also obtain $W_2$ because he knows how the decoder 2 works and able to apply this algorithm. By Information Theory, $nR_1 + nR_2 \leq n, \therefore R_1 + R_2 \leq 1$ for receiver 1. The encoder allocates time to transmitting respective messages. The outer bound is achieved in this way, a.k.a. time-sharing. Together with $R_2 \leq 1$ for receiver 2, the capacity region is the triangle with vertices (0,0), (0,1) and (1,0).

3. $W_{\mathcal{A}_1} = \emptyset$ and $W_{\mathcal{A}_2} = \emptyset$ None of the receivers has the side information as depicted in Figure 2.3.

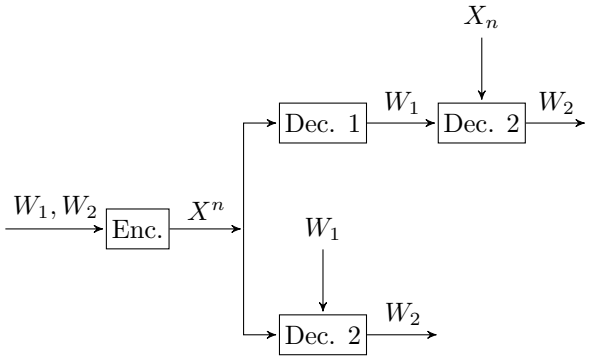



Figure 2.2: Ecoding scheme of $W_{\mathcal{A}_1} = \emptyset$ and $W_{\mathcal{A}_2} = W_1$

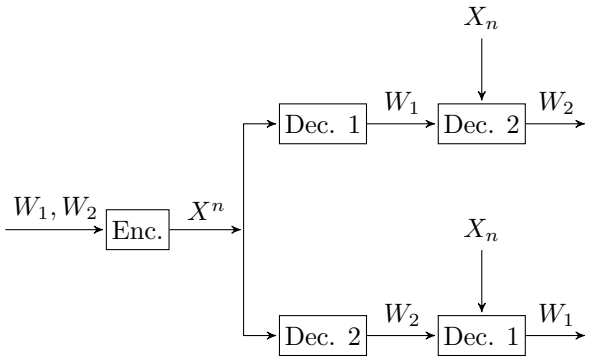



Figure 2.3: Encoding scheme of $W_{\mathcal{A}_1} = \emptyset$ and $W_{\mathcal{A}_2} = \emptyset$

For receiver 1 and 2, they both can recover $W_1$ and $W_2$. $R_1 + R_2 \leq 1$. Another way to deal with this case is that the network cannot outperform the case (2). The encoder achieves the outer bound also by time-sharing. The capacity region is also the triangle with vertices (0,0), (0,1) and (1,0).

We extend the case 1) in the 2-receiver discussion to consider a so-called *N-complete graph*.

**Definition 1.1.** In a network where $\mathcal{A}_j = [1 : N] \backslash \{j\}$, $\forall j$, the corresponding graph $\mathcal{G}$ is called a complete graph.

An example of $N = 4$ is shown in Figure 2.4. The encoder optimizes the transmission efficiency by **xor**ing $W_1, \ldots, W_N$. At receiver $j$, the decoder simply **xor**s $X^n$ and all of his prior side informations to recover $W_j$. Therefore, in N-complete graphs, $R_j = 1$, $\forall j$.

Because each receiver has its maximum prior side information, the other N-receiver networks cannot outperform this one. In a short summary, $R_j \leq 1$, $\forall j \in [1 : N]$, in any network.

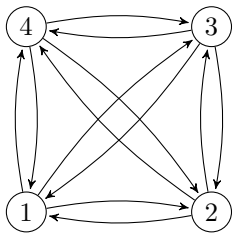



Figure 2.4: 4-complete graph

## 2.2 A Useful Outer Bound

Let me introduce a useful theorem in [4] which is useful in the later discussion for for gaining quick insight into the network.

**Theorem 2.1.** If rate tuple $(R_1, \ldots, R_N)$ is achievable for an Index Coding problem represented by the directed graph $\mathcal{G}$, then it must satisfy

$$\sum_{j \in \mathcal{J}} R_j \leq 1$$

for all $\mathcal{J} \subseteq [1 : N]$ for which no directed loops, or cycles, exist in the subgraph $\mathcal{G}$ over $\mathcal{J}$.

Take Figure 1.2 as an example. Receiver {1, 2} contains no directed loop and hence by Theorem 2.1, we have $R_1 + R_2 \leq 1$. Nevertheless, receiver {1, 3} contains a directed loop from receiver 1 to 3 and back, so we cannot say that $R_1 + R_3 \leq 1$.

I prove Theorem 2.1 in a similar thinking like what we do in 2-receiver case. With some graph features and the concept of information, we prove it and simultaneously reduce the complexity manipulation of Information Theory. First, consider the following claim.

**Claim.** If the subgraph $\mathcal{G}$ over $\mathcal{J}$, $\mathcal{J} \subseteq [1 : N]$, contains no directed loop, then there exists at least one receiver $p$, $p \in \mathcal{J}$, such that $\forall j \in \mathcal{J} \backslash \{p\}, j \notin \mathcal{A}_p$.

*Proof.* Without loss of generality (W.L.O.G.) , suppose $\mathcal{J} = [1 : k]$. If no receiver $p$ such that $\mathcal{A}_p = \emptyset$, that is, each one has at least one side information of others. Assume receiver 1 has $W_2$. Receiver 2 has also at least one side information. See Figure 2.5, it is clear that a directed loop forms if $1 \in \mathcal{A}_2$. Therefore, $1 \notin \mathcal{A}_2$. Let receiver 2 have $W_3$ and analyze similarly, we find that no matter whose messages is available at receiver $k$, a directed loop must appear in this subgraph. Contradiction! ($\Rightarrow\Leftarrow$) □

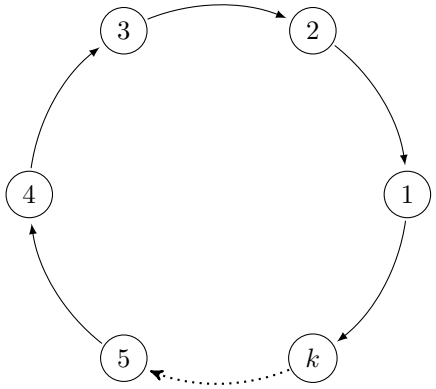



Figure 2.5: The graph for the proof of Claim

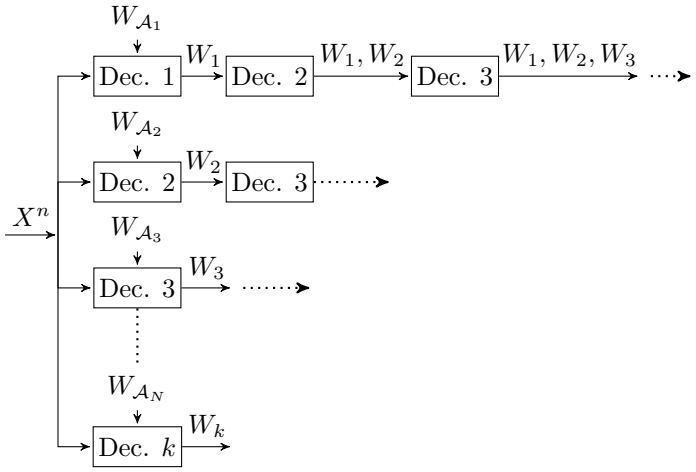



Figure 2.6: The decoding scheme in the proof of Theorem 2.1

Based on the claim, I prove Theorem 2.1.

*Proof.* Suppose that $\mathcal{J} = \mathcal{S}_1 = [1 : k]$ contains no directed loop and ,W.L.O.G. , receiver 1 has no side informations.

Consider $\mathcal{S}_2 = [2 : k]$. Since $\mathcal{S}_2 \subset \mathcal{S}_1$, implying that $\mathcal{S}_2$ contains no directed loop, there exists at least one receiver $p \in \mathcal{S}_2$ with no side informations of others in $\mathcal{S}_2$, and let $p = 2$. Notice that though receiver 1 do not have side informations, $W_1 \in \mathcal{A}_j$ , $j \in [2 : k]$ is allowed and possible.

Go on the process, we construct subsets $\mathcal{S}_1, \ldots, \mathcal{S}_k$. $\mathcal{A}_j \cap \mathcal{S}_{j+1} = \emptyset$ , $\forall j \in [1 : k]$ ($\mathcal{S}_{k+1} = \emptyset$) but $\mathcal{A}_j \subseteq \mathcal{S}_1 \backslash \mathcal{S}_j = [1 : j - 1]$. Look to Figure 2.6. Receiver 1 derives $W_1$ and then he can apply decoder 2 to obtain $W_2$. The rationale is that decoder 2 can obtain $W_2$ based on $X^n$ and $\mathcal{A}_2$, which possibly includes $W_1$.

After decoder $j$ , $j \in [1 : k - 1]$, receiver 1 has messages over [1: $j$]. Because $\mathcal{A}_{j+1} \subseteq [1 : j]$, he can continue to use decoder $j + 1$ to obtain $W_{j+1}$. In the end of deduction, receiver 1 derives $W_1, \ldots, W_k$. By Information Theory, we have

$$\sum_{j \in \mathcal{J}} R_j \leq 1 \tag{2.1}$$

□

### 2.2.1 $N$-directed Cycle

Now let's apply Theorem 2.1 on a simple family of directed graphs to see the benefit.

**Definition 2.2.** If $\mathcal{A}_j = \{((j + d) \bmod \mathrm{N}), d = \pm 1, \forall j \in [1 : N]$, then we say it forms a $N$-directed cycle.

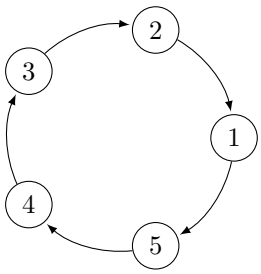



Figure 2.7: 5-directed Cycle

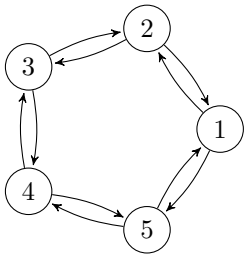



Figure 2.8: 5-receiver loop

Observe that $N-$ is the number of maximum receivers among which no directed loop forms. For distinct $j_k$'s

$$\sum_{k=1}^{N-1} R_{j_k} \leq 1, \tag{2.2}$$

We can simply derive that the sum rate, $\frac{N}{N-1}$ if rates equal by symmetry.

This sum rate can be achieved by transmitting a sequence of $W_{j_k,j_{k+1}} = W_{j_k}$ **xor** $W_{j_{k+1}}$, $k \in [1 : N-1]$. This sequence in fact implies $W_{j_1}$ **xor** $W_{j_N}$ by **xor**ing all the received messages. Each receiver just **xor**s and recovers the required message at rate $\frac{1}{N-1}$.

### 2.2.2 A Special Case and Symmetric $N$-receiver Loop

Consider the symmetric 5-receiver Index Coding problem as depicted in Figure 2.8.

We can use Theorem 2.1 to obtain outer bounds. However, paper [5] suggests the tighter inequality

$$R_1 + R_2 + R_3 + R_4 + R_5 \leq 2 \tag{2.3}$$

For convenience, I use the following terminology to describe the theorem in [5].

**Definition 2.3.** We say a relation $W_i \overset{k}{\longleftrightarrow} W_j$ iff $W_i \notin \mathcal{A}_k, W_j \notin \mathcal{A}_k$ for distinct $i$, $j$, $k \in \mathcal{V}$.

**Theorem 2.2.** For any $N+1$ distinct indices $i_0$,$i_1$, ..., $i_N \in \mathcal{V}$ and $N$ indices $m_1$,..., $m_N \in \mathcal{V}$ if there is the following alignment chain

$$W_{i_0} \overset{j_1}{\longleftrightarrow} \ldots \overset{j_{N-1}}{\longleftrightarrow} W_{i_{N-1}} \overset{j_N}{\longleftrightarrow} W_N, W_{i_0} \notin \mathcal{A}_{i_N} \tag{2.4}$$

then,

$$\sum_{m=0}^{N} R_{i_m} + \sum_{l=1}^{N} R_{j_l} \leq N \tag{2.5}$$

I omit the proof which requires Information Theory and Fano's inequality. It can be found in [5].

Surprised by the Theorem 2.2, I investigate the general symmetric $N$-receiver loop.

**Definition 2.4.** If $\mathcal{A}_j = \{((j+d) \bmod \mathrm{N}) \, , d = \pm 1\}$ , $\forall j \in [1 : N]$ , then we say it forms a symmetric $N$-receiver loop.

The cases $N = 1$, 2 ,3 in fact are complete graphs. I in turn analyze the cases $N \geq 4$. I apply the two proposed and proved outer bounds to the symmetric N-receiver loops. Since the graph is symmetric, I still focus on the sum rates. I categorize these graphs by the oddity of $N$.

#### Even Receivers

Apply Theorem 2.1 to $j = 2k-1$ , $k \in [1 : N/2]$ because it contains no directed loop. Then,

$$R_1 + R_3 + \ldots + R_{\frac{N}{2}-1} \leq 1 \tag{2.6}$$

Similarly,

$$R_2 + R_4 + \ldots + R_{\frac{N}{2}} \leq 1 \tag{2.7}$$


(a) Symmetric 7-receiver loop (b) Improved network of the symmetric 7-receiver loop

Figure 2.9: Improved network of the symmetric 7-receiver loop

Combine the inequalities (2.6) and (2.7), we have

$$\sum_{j=1}^{N} R_j \leq 2 \tag{2.8}$$

To reach the outer bound in (2.8), consider the following coding scheme

**Definition 2.5.** The sender divides the time into $N$ slots. In $j$-th slot, the encoder transmits $W_j$ **xor** $W_{j+1}$ ($N+1=1$). The $j$-th receiver also applies **xor**ing, at rate $R_j = 2/N$. I name this coding method as *XOR coding cycle*.

**Odd Receivers**

It is more complicated in this case. The maximum receivers of no directed loop is $\frac{N-1}{2}$ Again, apply Theorem 2.1 to $j = 2k-1$ , $k \in [1 : \frac{N-1}{2}]$,

$$R_1 + R_3 + \ldots + R_{N-2} \leq 1 \tag{2.9}$$

We can write down (2.9) by starting from different indices, but, given symmetry, it is more convenient to average 1 by the number of receivers on the left of (2.9) and multiply by $N$ to reach the sum rate. Hence,

$$\sum_{j=1}^{N} R_j \leq (\frac{1}{\frac{N-1}{2}} \times N) = 2 + \frac{2}{N-1} \tag{2.10}$$

In comparison, we can replace the $N$ in Theorem 2.2 with $\frac{N-1}{2}$ and set $i_0 = 1$, $i_1 = 2$, $\ldots$,$i_{\frac{N-1}{2}} = \frac{N+1}{2}$, $j_1 = \frac{N+3}{2}$, $\ldots$, $j_{\frac{N-1}{2}} = N$. Then,

$$\sum_{j=1}^{N} R_j \leq \frac{N-1}{2} \tag{2.11}$$

It is obvious that (2.11) is not a useful outer bound since it approaches infinity as $N$ is sufficiently large.

By XOR coding cycle, we achieve (2.3) and (2.10). We see that as $N$ is sufficiently large, the sum rate approach to 2, regardless of oddity of $N$. If fact, for even users, the sum rate is exactly 2.

With the knowledge of the case $N = 5$, I introduce another system. See Figure 2.9a and 2.9 for case $N = 7$.

Now I define the node such as (4, 6) in Figure 2.9.

**Definition 2.6.** Multiple receivers $j_1$, $j_2$, $\ldots$, $j_K$,comprise a single receiver system. The sender transmits $W_{system}$ which is the sequence of corresponding messages $W_1$, $W_2$, $\ldots$,$W_K$. The system rate $R_{system} = \sum_{m=1}^{K} R_{j_m}$.

I improve the symmetric 7-receiver loop by letting the system of receivers 4, 6 know $W_3$, $W_5$, $W_7$ and that of receivers 5, 7 know $W_1$, $W_4$, $W_6$. It becomes a symmetric 5-receiver loop, and by (2.3)

$$R_1 + R_2 + R_3 + R_{system\{4,6\}} + R_{system\{5,7\}} \leq 2 \tag{2.12}$$

By Definition 2.6,

$$R_1 + R_2 + R_3 + R_4 + R_5 + R_6 + R_7 \leq 2 \tag{2.13}$$

Since the network of symmetric 7-receiver loop cannot outperform the improved one, (2.13) is valid for the original network. The sum rate for $N = 7$ is 2 and is achieved by applying XOR coding cycle.

The analysis is the same for all odd receivers, and the sum rates are all equal to 2. In summary, the sum rates in cases $N \leq 3$ are $1 \times N = N$ while equal to 2 for $N \geq 4$.

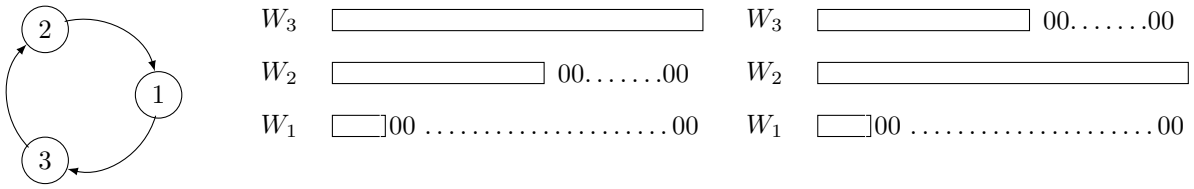
## 2.3 General 3-directed Cycle

In 3-directed cycle as depicted in Figure 2.10, I assume that $W_j$ are of equal length and hence the same rate $R_j$. Nevertheless, in general $W_j$ are unequal. I in turn propose the following coding scheme to achieve the outer bound in Theorem 2.1 . The outer bounds are

$$R_1 + R_2 \leq 1 \tag{2.14}$$

$$R_2 + R_3 \leq 1 \tag{2.15}$$

$$R_3 + R_1 \leq 1 \tag{2.16}$$




(a) 3-directed cycle (b) $R_3 \geq R_2 \geq R_1$ (c) $R_2 \geq R_3 \geq R_1$

Figure 2.10: 3 receivers form a directed cycle

In Figure 2.10b, add consecutive 0's behind $W_2$, and **xor** with $W_3$, and say it $W_{2,3}$. Similarly, add consecutive 0's behind $W_1$, and **xor** with $W_2$, and say it $W_{1,2}$. The sender broadcasts these two encoded messages. The decoder 1, who knows in advance the encoding algorithm, obtains $W_1$ by eliminating 0's after **xor**ing $W_{1,2}$ and $W_2$. So does the decoder 2. As for the decoder 3, he **xor**s $W_{1,2}$ and $W_{2,3}$, obtains $W_{1,3}$ and thus $W_3$.

In this coding scheme, the sender totally broadcasts the bits $n$ as the total length of $W_2$ and $W_3$, that is, $R_2 + R_3 = 1$. Since $R_1 \leq R_2, R_3$, the capacity region formed by (2.14)-(2.16) is achieved.

As the case with Figure 2.10b, I deal with the situation in Figure 2.10c in the same manner. The sender transmits the **xor**ed results of the more lengthy messages to achieve the capacity region.

# Chapter 3

# Heterogeneous Caching Memory Allocation

So far, we consider the sum rate in the case of passive server. However, if we can design which side information each user has by caching in advance, how can we exploit the inherent coding opportunities in caching and requested files to lower the rate of sending files over the congested network?

For this case of active server, we have seen the problem description in chapter 1. Paper [6] proposes an centralized algorithm for system where users all have the same uniform preference distribution and prove that the ratio of upper bound of the worst-case rate to lower cut-set bound is within a constant regardless of parameters.

They consider the nonuniform preference distribution in [6]. They show that the ratio of upper bound of the rate to lower cut-set bound is within a constant times the number of file partitions.

Here we study the scenario where all users have the same uniform preference distribution but they form two groups, the basic case where heterogeneous cache sizes exist,with different cache memories.

I first try the decentralized caching scheme and then centralized one. To evaluate the cut-set lower bound in this system, I extend the cut-set bound in [6] in a similar way.

Consistent to notations, we have $N$ files in total and all the users have the same uniform access distribution to these files. Two groups of users $\mathcal{K}_1, \mathcal{K}_2$, and the group members are $|\mathcal{K}_1| = K_1, |\mathcal{K}_2| = K_2$ respectively. Within each group $\mathcal{K}_i, i = 1, 2$, the members are each equally equipped with cache memory size $M_i, i = 1, 2$.

## 3.1 Cut-set Bound

In the paper [6], they propose the following lower bound for the scenario where all users are equipped the same memory size.

$$R^*(M) \geq \max_{s \in [1:\min\{N,K\}]} \left(s - \frac{s}{\lfloor N/s \rfloor} M\right) \tag{3.1}$$

In a similar manner, we can draw $s_i$ members from group $\mathcal{K}_i, i = 1, 2$, and construct the cut-set lower bound on rate $R^*(K_1, K_2, M_1, M_2)$. Denote $s = (s_1, s_2)$. The maximum is found over the region $s \in \mathcal{R}_s$.

$$R^*(K_1, K_2, M_1, M_2, N) \geq \max_{s \in \mathcal{R}_s} \left((s_1 + s_2) - \frac{s_1 M_1 + s_2 M_2}{\lfloor N/(s_1 + s_2) \rfloor}\right) \tag{3.2}$$

The region $\mathcal{R}_s$ depends on relation of $N, K_1, K_2$, The main principle is that the members $s_i$ you draw from group $\mathcal{K}_i$ is limited by $K_i$. The total members you pick can neither excess $N$ nor equal zero so that we are able to allocate files to selected members and use cut-set bound. I list the four cases when there are two groups.

1. $N \geq K_1 + K_2$

   $\mathcal{R}_s = \{s_1 \in [0 : K_1], s_2 \in [0 : K_1], s_1 + s_2 \in [1 : K_1 + K_2]\}$

2. $K_1 + K_2 \geq N \geq \max\{K_1, K_2\}$

   $\mathcal{R}_s = \{s_1 \in [0 : K_1], s_2 \in [0 : K_1], s_1 + s_2 \in [1 : N]\}$

3. $\max\{K_1, K_2\} \geq N \geq \min\{K_1, K_2\}$

   $\mathcal{R}_s = \{s_1 \in [0:N], s_2 \in [0:K_2], s_1 + s_2 \in [1:N]\}$

4. $\min\{K_1, K_2\} \geq N$

   $\mathcal{R}_s = \{s_1 \in [0:N], s_2 \in [0:N], s_1 + s_2 \in [1:N]\}$

The result can be extended to $G$ groups of users. Let $s = (s_1, s_2, \ldots, s_G)$

$$R^*(M_1, M_2, \ldots, M_G, K_1, K_2, \ldots, K_G, N) \geq \max_{s \in \mathcal{R}_s} \Big( \sum s_i - \frac{\sum s_i M_i}{\lfloor N / \sum s_i \rfloor} \Big) \tag{3.3}$$

For any subset $\mathcal{S} \subset [1:G]$.

$$\mathcal{R}_s = \begin{cases} |\mathcal{S}| = 1: & s_i \in [0 : \min\{N, K_i\}], \forall i, \\ |\mathcal{S}| = 2: & \sum_{i \in \mathcal{S}} s_i \in [0 : \min\{N, \sum_{i \in \mathcal{S}} K_i\}], \\ \ldots & \\ |\mathcal{S}| = G-1: & \sum_{i \in \mathcal{S}} s_i \in [0 : \min\{N, \sum_{i \in \mathcal{S}} K_i]\}, \\ |\mathcal{S}| = G: & \sum_{i \in \mathcal{S}} s_i \in [1 : \min\{N, \sum K_i\}] \end{cases}$$

## 3.2 Achievable Rate by Decentralized Caching Scheme

There are two ways of placement phase, based on files or users. Since both $\mathcal{K}_1, \mathcal{K}_2$ have uniform preference over $N$ files, user belonging to $\mathcal{K}_i$ allocates $M_i/N$ file size of memory for caching each file. I denote $R(K_1, K_2, M_1, M_2, N)$. To simplify notations, I make $M_1/N = q_1, M_2/N = q_2$.

**File-based**

If $K_1 + K_2 > N$, the worst case is that at least one user requires every file. Each user of $\mathcal{K}_i$ requesting some file has already $Fq_i$ bits, for large $F$, at most

$$F(1 - q_i) + o(F) \tag{3.4}$$

must be sent by the server. I will ignore $o(F)$ in the following by assuming $F$ is large. Since there are $N$ files, and we can compute the upper bound.

$$F(1 - \min\{q_1, q_2\})N \tag{3.5}$$

On the other hand, if $K_1 + K_2 \leq N$ The worst case is that at most $K_1 + K_2$ different files are requested, with different users requesting different files.

$$F(1 - q_1)K_1 + F(1 - q_2)K_2 \tag{3.6}$$

**User-based**

In the paper [8], $V_{k,\mathcal{S}}$ means the bits of the file $d_k$ requested by user $k$ cached exclusively at users in $\mathcal{S}$. I follow this notation. A bit of file $d_k$ is in $V_k, \mathcal{S}$ if it's present in the cache of every user in $, \mathcal{S}$ and absent from the cache of every user outside $\mathcal{S}$. Let's consider a subset $\mathcal{S} \subset \mathcal{K}$, with $|\mathcal{S}| = s$. It is composed of $t_1$ of $\mathcal{K}_1$ users and $t_2$ of $\mathcal{K}_2$ users, so $t_1 + t_2 = s$ The expected size of $V_{k,\mathcal{S}\setminus\{k\}}$, if $k \in \mathcal{K}_1$, is

$$Fq_1^{t_1 - 1} q_2^{t_2} (1 - q_1)^{K_1 - t_1 + 1} (1 - q_2)^{K_2 - t_2} \tag{3.7}$$

Otherwise, if $k \in \mathcal{K}_2$, is

$$Fq_1^{t_1} q_2^{t_2 - 1} (1 - q_1)^{K_1 - t_1} (1 - q_2)^{K_2 - t_2 + 1} \tag{3.8}$$

If $t_1 t_2 \neq 0$, for this subset $\mathcal{S}$, send the maximum of $V_{k,\mathcal{S}\setminus\{k\}}$

$$\max_{k \in \mathcal{S}} \quad V_{k,\mathcal{S}\setminus\{k\}} = Fq_1^{t_1} q_2^{t_2} (1 - q_1)^{K_1 - t_1} (1 - q_2)^{K_2 - t_2} \max\{\frac{1 - q_1}{q_1}, \frac{1 - q_2}{q_2}\} \tag{3.9}$$

If $t_2 = 0$, send (9), while for $t_2 = 0$, send (10). Then we are to sum all the possible composition of $(t_1, t_2)$ with both greater than zero. But the technique here is to include the both-zero case, and them subtract it from the result.

$$\begin{aligned}
&\sum_{t_1=1}^{K_1}\sum_{t_2=1}^{K_2}\binom{K_1}{t_1}\binom{K_2}{t_2}Fq_1^{t_1}q_2^{t_2}(1-q_1)^{K_1-t_1}(1-q_2)^{K_2-t_2}\max\{\frac{1-q_1}{q_1},\frac{1-q_2}{q_2}\}\\
=&F\max\{\frac{1-q_1}{q_1},\frac{1-q_2}{q_2}\}\sum_{t_1=1}^{K_1}\binom{K_1}{t_1}q_1^{t_1}(1-q_1)^{K_1-t_1}\sum_{t_2=1}^{K_2}\binom{K_2}{t_2}q_2^{t_2}(1-q_2)^{K_2-t_2} \qquad (3.10)\\
=&F\max\{\frac{1-q_1}{q_1},\frac{1-q_2}{q_2}\}\big[1-(1-q_1)^{K_1}\big]\big[1-(1-q_2)^{K_2}\big]
\end{aligned}$$

As $t_1 = 0$, send

$$F\frac{1-q_2}{q_2}(1-q_1)^{K_1}(1-(1-q_2)^{K_2}) \qquad (3.11)$$

As $t_2 = 0$, send

$$F\frac{1-q_1}{q_1}(1-q_2)^{K_2}(1-(1-q_1)^{K_1}) \qquad (3.12)$$

It yields to send totally

$$\begin{aligned}
&F\max\{\frac{1-q_1}{q_1},\frac{1-q_2}{q_2}\}\big[1-(1-q_1)^{K_1}\big]\big[1-(1-q_2)^{K_2}\big]\\
+&F\frac{1-q_2}{q_2}(1-q_1)^{K_1}(1-(1-q_2)^{K_2}) \qquad (3.13)\\
+&F\frac{1-q_1}{q_1}(1-q_2)^{K_2}(1-(1-q_1)^{K_1})
\end{aligned}$$

$R^*(K_1, K_2, M_1, M_2, N)$ is upper bounded by the minimum of equation (3.5),(3.6),(3.13).

## 3.3 Margin between Bounds in Decentralized Scheme

We have established upper bound on expected worst-case rate $R$ in two groups by using file-based and user-based methods. $(q_1 = M_1/N, 1 = M_2/N)$

$$R^* \le \min\left\{\begin{array}{l}\max\{1-q_1, 1-q_2\}N,\\ (1-q_1)K_1+(1-q_2)K_2,\\ \max\{\frac{1-q_1}{q_1},\frac{1-q_2}{q_2}\}[1-(1-q_1)^{K_1}(1-q_2)^{K_2}]\end{array}\right\}$$

Recall the lower bound,

$$R^* \ge \max_{s_1,s_2}\Big(s_1+s_2-\frac{s_1M_1+s_2M_2}{\lfloor\frac{N}{s_1+s_2}\rfloor}\Big) \qquad (3.14)$$

To make matters clear, I let $\overline{R}$ denote the upper bound on $R$, and $\underline{R}$ lower bounds $R$, that is, $\overline{R} \ge R^* \ge \underline{R}$.

I run the simulation to see if the margin between $\overline{R}$ and $\underline{R}$ is universal without regard to $M_1, M_2, K_1, K_2, N$. It turns out that the margin is seemingly not universal under decentralized scheme. In fact, it is not. To see this, consider $K_1 << K_2$, and

$$\begin{aligned}
&M_1/N \le 1/(K_1+K_2)\\
&\tfrac{1}{K_1+1} \le M_2/N \le \tfrac{3/2}{K_1+1}
\end{aligned}$$

Then derive the bounds,

$$\begin{aligned}
&\overline{R} = K_1+K_2-\frac{K_1M_1+K_2M_2}{N} \ge K_1+K_2-\frac{K_1}{K_1+K_2}-\frac{3K_2/2}{K_1+1}\\
&\underline{R} = K_1-\frac{K_1M_1}{\lfloor N/K_1\rfloor} \le K_1-\frac{K_1^2M_1}{N} \le K_1 \qquad (3.15)\\
&\Rightarrow \frac{\overline{R}}{\underline{R}} \ge 1-\frac{1}{K_1+K_2}+\frac{K_2}{K_1}\frac{K_1-1/2}{K_1+1} \ge \frac{K_2}{4K_1}
\end{aligned}$$

The lower bound occurs at $s = K_1$, because we can first restate the lower bound if $M_1 \leq M_2$.

$$\underline{R} \geq \max_{s\in[1,K_1+K_2]} s - \frac{\min\{s, K_1\}M_1 + \lfloor s - K_1 \rfloor_+ M_2}{\lfloor N/s \rfloor} \tag{3.16}$$

The settings on $M_1, M_2$ satisfies this condition. If $s \leq K_1$, the RHS of (3.16) has its maximum at $s = K_1$ by checking it is a parabola and vertex excess $K_1$. As $s$ excesses $K_1$, RHS becomes smaller, for $t \in [1, K_2]$

$$[(K_1 + t) - \frac{K_1M_1 + tM_2}{\lfloor N/(K_1 + 1) \rfloor}] - [K_1 - \frac{K_1M_1}{\lfloor N/K_1 \rfloor}] \leq t\Big(1 - \frac{(K_1 + 1)M_2}{N}\Big) \leq 0 \tag{3.17}$$

We can set constraints on the parameters and then follow the derivation in [7]. I introduce some constraints on the ratio of $K_1$ to $K_2$, $N$ to $K_1M_1$, $N$ to $K_2M_2$. I call $K_iM_i$ the group cache memory.

$$\min\{N, K_1 + K_2\} \leq C_1 \tag{3.18}$$

$$\frac{1}{C_2 - 1} \leq \frac{K_1}{K_2} \leq C_2 - 1 \tag{3.19}$$

$$\frac{N}{K_iM_i} \leq C_3, \forall i \tag{3.20}$$

With these three constraints, one can make sure the margin become universally bounded by a function of $C_1, C_2, C_3$. W.L.O.G., we assume $M_1 \leq M_2 \leq N$ so $\overline{R}$ can be bounded by

$$\overline{R} \leq (1 - q_1) \min\{\frac{1}{q_1}, N, K_1 + K_2\} \tag{3.21}$$

Let's follow the discussion in paper. For $0 \leq \min\{N, K_1 + K_2\} \leq C_1$,

$$\overline{R} \leq (1 - q_1) \min\{N, K_1 + K_2\} \leq (1 - q_1)C_1 \tag{3.22a}$$

$$\underline{R} \geq 1 - q_1 (s_1 = 1, s_2 = 0) \tag{3.22b}$$

$$\Rightarrow \frac{\overline{R}}{\underline{R}} \leq C_1 \tag{3.22c}$$

Assume $\min\{N, K_1 + K_2\} \geq C_1 + 1$, consider the following cases,

$$M \in \begin{cases} [0, \max\{1, N/(K_1 + K_2)\}] \\ (\max\{1, N/(K_1 + K_2)\}, N/C_1] \\ (N/C_1, N] \end{cases}$$

For first case, $M \in [0, \max\{1, N/(K_1 + K_2)\}]$,

$$\overline{R} \leq (1 - q_1) \min\{N, K_1 + K_2\} \leq \min\{N, K_1 + K_2\} \tag{3.23a}$$

$$\underline{R} \geq \min\{N, K_1 + K_2\}(1/C_2 - 1/(C_1 + 1) - \frac{1/C_2^2}{1 - 1/C_2}) \tag{3.23b}$$

$$\Rightarrow \frac{\overline{R}}{\underline{R}} \leq \frac{1}{1/C_2 - 1/C_1 - \frac{1/C_2^2}{1-1/C_2}} \tag{3.23c}$$

Here I set $s_1 = \lfloor \min\{N, K_1+K_2\}/C_2 \rfloor, s_2 = 0$, and it's valid since $s_1 = \lfloor \min\{N, K_1+K_2\}/C_2 \rfloor \in [1, \min\{N, K_1\}]$.

For second case, $\max\{1, N/(K_1 + K_2)\} < M \leq N/C_1$,

$$\overline{R} \leq (1 - q_1)\frac{1}{q_1} \leq q_1 = \frac{N}{M_1} \tag{3.24a}$$

$$\underline{R} \geq \frac{N}{M_1}(1/C_3 - 1/C_1 - \frac{1/C_3^2}{1 - 1/C_3}) \tag{3.24b}$$

$$\Rightarrow \frac{\overline{R}}{\underline{R}} \leq \frac{1}{1/C_3 - 1/C_1 - \frac{1/C_3^2}{1-1/C_3}} \tag{3.24c}$$

Here I set $s_1 = \lfloor N/(C_3M) \rfloor, s_2 = 0$, and it's valid since $s_1 = \lfloor N/(C_3M) \rfloor \in [1, \min\{N, K_1\}]$.

For the last one $N/C_1 < M \leq N$,

$$\overline{R} \leq (1-q_1)\frac{1}{q_1} = N/M_1 - 1 \tag{3.25a}$$

$$\underline{R} \geq 1 - M_1/N (s_1 = 1, s_2 = 0) \tag{3.25b}$$

$$\Rightarrow \frac{\overline{R}}{\underline{R}} \leq N/M_1 \leq C_1 \tag{3.25c}$$

For $M_2 \leq M_1 \leq N$, the analysis is similar by interchanging the role of 1 and 2. We can get the same result, that is,

$$\frac{\overline{R}}{\underline{R}} \leq \min\left\{C_1, \frac{1}{1/C_2 - 1/(C_1+1) - \frac{1/C_2^2}{1-1/C_2}}, \frac{1}{1/C_3 - 1/C_1 - \frac{1/C_3^2}{1-1/C_3}}\right\} \tag{3.26a}$$

Thus, we can manipulate $C_1, C_2, C_3$ to obtain a favorite universal bound on the margin.

## 3.4 Margin between Bounds in Centralized Scheme

Now that the decentralized caching scheme does not lead to universal margin, I apply the centralized one to see if the margin can be universally good. Recall from the paper [6], the achievable rate of a system with $M, K, N$ is

$$\overline{R} = K(1-\frac{M}{N})\min\{\frac{1}{1+\frac{KM}{N}}, \frac{N}{K}\} \tag{3.27}$$

where $M = \frac{iN}{K}, i = 1, 2, \ldots, K-1$. I address the group with different caching sizes separately, that is, use centralized schemes on each group. $M_i = \frac{jN}{K}, i = 1, 2$ and $j = 1, 2, \ldots, K_i - 1$. Without loss of generosity, suppose $M_1, M_2 < N$. The lower bound is,

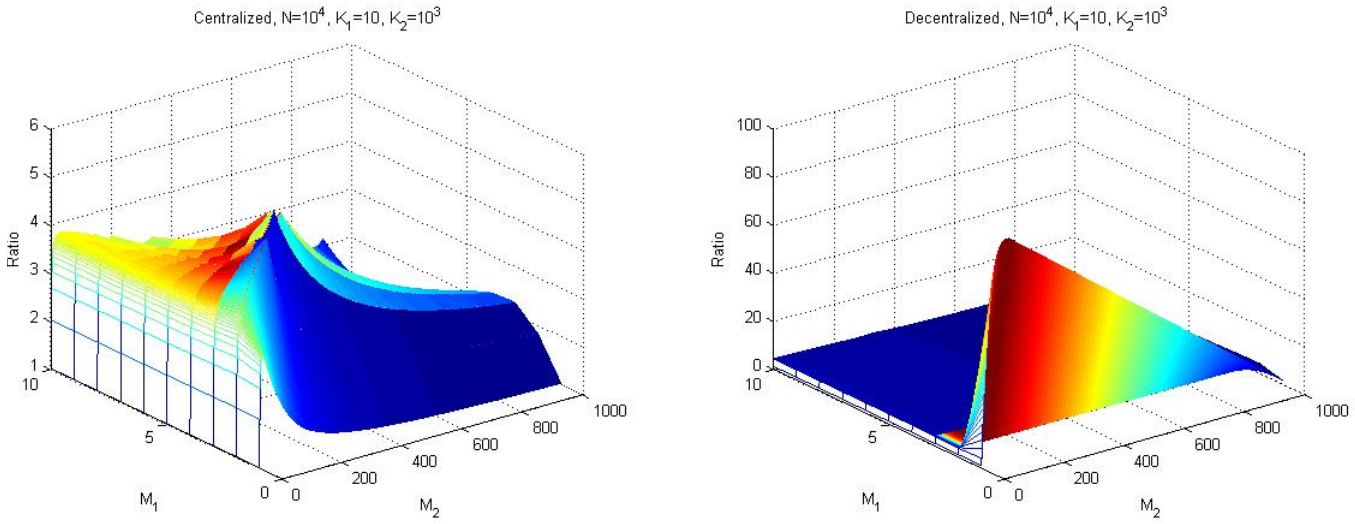
$$\underline{R} \geq \max_{s\in[1,K_1+K_2]} s - \frac{\min\{s, K_1\}M_1 + \lfloor s - K_1 \rfloor_+ M_2}{\lfloor N/s \rfloor} \tag{3.28}$$

It turns out that the margin is universally bounded by 8.

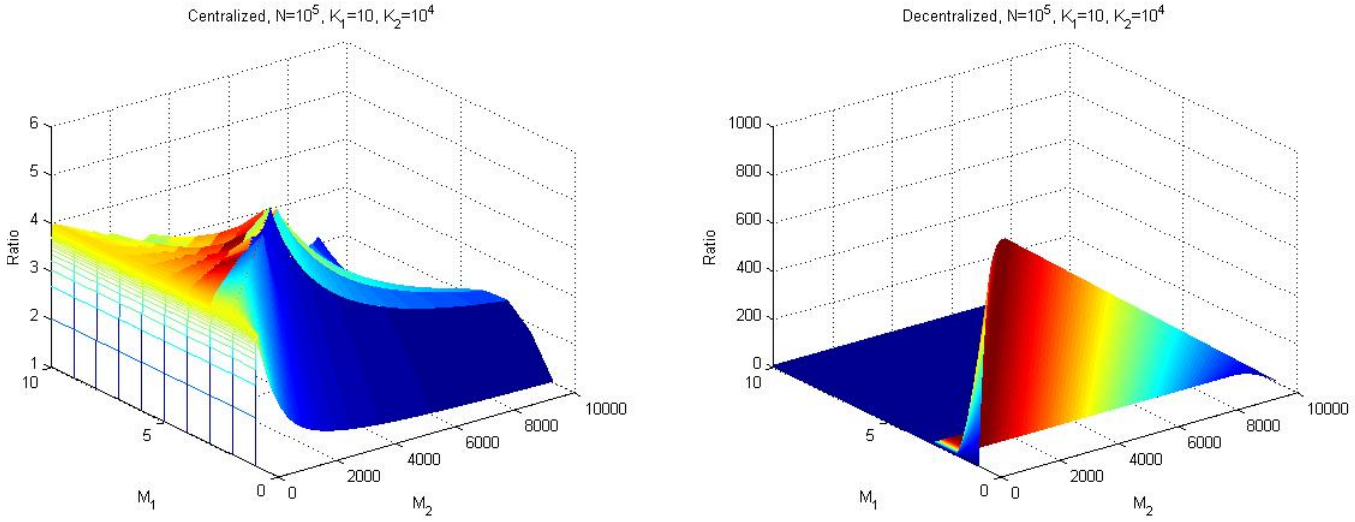
$$\begin{aligned}
\overline{R} &= K_1(1-\frac{M_1}{N})\min\{\frac{1}{1+\frac{K_1M_1}{N}}, \frac{N}{K_1}\} + K_2(1-\frac{M_2}{N})\min\{\frac{1}{1+\frac{K_2M_2}{N}}, \frac{N}{K_2}\} \\
&\leq K_1(1-\frac{M_1}{N})\frac{1}{1+\frac{K_1M_1}{N}} + K_2(1-\frac{M_2}{N})\frac{1}{1+\frac{K_2M_2}{N}} \\
\underline{R} &\geq \frac{N}{2M_1} - \frac{N^2}{4M_1^2}\frac{M_1}{N} = \frac{N}{4M_1} (s = \lceil \frac{N}{2M_1} \rceil) \\
\Rightarrow \frac{\overline{R}}{\underline{R}} &\leq \frac{4M_1}{N}\Big(\frac{K_1}{1+\frac{K_1M_1}{N}} + \frac{K_2}{1+\frac{K_2M_2}{N}}\Big) \leq 4(1+\frac{M_1}{M_2}) \leq 8
\end{aligned} \tag{3.29}$$

For multiple groups, again we can rank the memory size and the lower bound still depends on the least one. The upper bound is added another term, and the inequality for bounding margin thus consider another term that is small than 4. Therefore, the bound on margin increases if the number of groups grows.

I plot the margins under centralized and decentralized caching schemes given two sets of $N, K_1, K_2$ over all possible $M_1, M_2$ in Figure 3.1. One can observe that with $K_2$ is increased to ten times, the margins also follow in decentralized scheme. This matches the inequality in (3.15). The centralized method, nevertheless, generates margins lower than 6, which corresponds to the result in (3.29).




(a) $N = 10^4, K_1 = 10, K_2 = 1000$




(b) $N = 10^5, K_1 = 10, K_2 = 1000$

Figure 3.1: Comparisons of margins under centralized and decentralized caching schemes given $N, K_1, K_2$

# Chapter 4

# Heterogeneous Preference Distribution

Our major interest is on the scenario where access distributions over the files of each user differ. It is the heterogeneous preference distribution case. If the cache memory size are the same over all the users, we expect that each user may allocate different amount of cache memory given one arbitrary file. Hence in the placement phase, we need to take into account the allocation of memory to each files which is the same in the case of uniform demand we discuss before.

Not to be complicated by the multiple users having same preference in a group, I explore users instead of groups, in fact one user in one group, for finding the coding opportunities and the scheme of memory allocation. I investigate two caching schemes on two user case with particular symmetric demands. Then I generalize the caching scheme and find the optimal solution by MATLAB. Eventually, I derive the general caching in three-user case.

Let me introduce the scenario and notation. Two users, say $K = 2$, want to access one of $N$ files in the server and $N \geq 2$. Each user is equipped with cache of size $F$ bits, that is, $M = 1$. I assume user $k$ has the largest probability $p$ of accessing the $k$ file and uniform over the other $N-1$ files, or probability $q = \frac{1-p}{N-1}$, and $p \geq q$. For example when $K = 2, N = 4$.

$$\mathbf{A} = \begin{pmatrix} p & q & q & q \\ q & p & q & q \end{pmatrix}$$

In the following, I propose two intuitive caching schemes: proportional to preference caching and complete files caching. In the former, we allocate the cache in the same proportion to access distribution, while we directly allocate all the cache to the file with highest access probability in the later.

## 4.1 Proportional to Preference Caching

Each user allocates its cache to each file according to its access distribution and randomly caches the bits. More specifically, $\mathbf{A} = \mathbf{C}$. If user $k$ will access file $n$ with probability $p$, it has fraction $Mp$ of file $n$ in its cache. Here I suppose $M = 1$ so the probability of some file also means the fraction of it that is cached.

I divide the scenarios into two: access the same or different files. After analyzing the scenarios, I take probabilities into account and derive the expected transmission measured in file.

#### Access the Same File

Two users access the same file.

Suppose user 1 caches $x$ fraction of the file, user 2 caches $y$ fraction, and they have the same cached $s$ fraction ($s \leq \min\{x, y\}$) that do not need transmitting. First, the server do not transmit the same $s$ fraction. Second, the server can perform **xor** on $x - s$ and $y - s$ fraction of files confined to which is smaller. The server still needs to send remaining bits that cannot be **xor**ed but exist in one's cache. Combining these, the server needs to transmit $\max\{x - s, y - s\}$.

Finally, it sends bits that both users do not cache. Therefore, the server totally sends

$$R = \max\{x - s, y - s\} + \{1 - (x + y - s)\} = 1 - \min\{x, y\} \tag{4.1}$$

In my assumed access distribution, $x, y = \{p, q\}$, but actually, $1 - \min\{x, y\} = 1 - q$ is the only result because $x$ and $y$ can not both be $p$.

**Access Different Files**

Two users access different files.

There are five coding cases in which two users access different files. I focus on the bits lengths that are not cached since these are what the server needs to transmit.

- User 1 caches file 1 and user 2 caches file 2:

  Both users access the $p$-files. The fraction can be coded for both users is $(1-p)q$. They can be transmitted one time by **xor** without remaining bits. The server sends the remaining bits $(1 - p)(1 - q)$ that exist neither in user 1's cache nor user 2's. Since now accessed files are different, we have to send these fraction two times.

  $$R = (1 - p)q + 2 \times (1 - p)(1 - q) = (1 - p)(2 - q) \tag{4.2}$$

- User 1 accesses file 2 and user 2 caches file 1:

  This case is symmetric to the previous one, exchange $p, q$ and we have

  $$R = (1 - q)p + 2 \times (1 - q)(1 - p) = (1 - q)(2 - p) \tag{4.3}$$

- One accesses file 1 or 2 and the other accesses file $k \geq 3$:

  The one accessing file $k \geq 3$ has $(1 - q)q$ that can be coded while $(1 - q)^2$ to be sent.

  If the other access the file with probability $p$, it has $(1 - p)q$ that can be coded. According to previous discussion, needed transmission is $\max\{(1 - q)q, (1 - p)q\}$ in this step. Finally, the server sends those bits that are not coded: $(1 - q)^2$ and $(1 - p)(1 - q)$.

  $$\begin{aligned} R =& \max\{(1 - q)q, (1 - p)q\} + (1 - q)^2 + (1 - p)(1 - q) && \text{(4.4a)} \\ =& (1 - q)(2 - p) && \text{(4.4b)} \end{aligned}$$

  If the other one wants the file with probability $q$, it has $(1 - q)p$ that can be coded. Similarly, we can derive

  $$\begin{aligned} R =& \max\{(1 - q)q, (1 - q)p\} + (1 - q)^2 + (1 - q)(1 - p) && \text{(4.5a)} \\ =& (1 - q)(2 - q) && \text{(4.5b)} \end{aligned}$$

- Both access files $k \geq 3$:

  The fraction can be coded for both users is $(1 - q)q$. However, we need to send two times the fraction $(1 - q)^2$.

  $$(1 - q)q + 2 \times (1 - q)^2 = (1 - q)(2 - q) \tag{4.6}$$

**Expected Transmission**

Considering probability of each case, I derive the expression of expected transmission.

$$\begin{aligned} E[R] =& [2pq + (N - 2)q^2][1 - q] && \text{(4.7a)} \\ &+ [p^2][(1 - p)(2 - q)] && \text{(4.7b)} \\ &+ [q^2][(1 - q)(2 - p)] && \text{(4.7c)} \\ &+ [2(N - 2)pq][(1 - q)(2 - p)] && \text{(4.7d)} \\ &+ [2(N - 2)q^2][(1 - q)(2 - q)] && \text{(4.7e)} \\ &+ [(N - 2)(N - 3)q^2][(1 - q)(2 - q)] && \text{(4.7f)} \end{aligned}$$

The former square brackets means the probability while the latter is the needed transmission in each case. I do not simplify (4.8) because it is a tedious work and this expression shows how it is derived. Later I use MATLAB to draw $E[R]$ versus $p$.

## 4.2 Complete Files Caching

Now the server stores complete files in each user in advance and confined to $M = 1$, it intuitively caches the file with the largest probability $p$.

#### Access the Same File

Once two users access the same file, the server cannot perform **XOR** coding but can send just one file to satisfy the requests since the users do not cache the same file.

#### Access Different Files

In each possible situation, the server at most sends two files. I explore the case where traffic can be reduced.

- Coding:

  The only case where coding can be performed is that user 1 requests file 2 and user 2 requests file 1. The server sends the files which results from bit-wise **XOR** on file 1 and 2, hence $R = 1$.

- Caching:

  If anyone accesses the cached file, $R$ can be reduced by one. Except for the cases of accessing the same file, there is one case ($p^2$) with $R = 0$ and $2(N-2)$ cases (pairs of $p$ and $q$) with $R = 1$ . The server is required to send two files, $R = 2$, in the other cases.

#### Expected Transmission

Taking probability of each case into account, I derive the expected transmission.

$$\begin{aligned}
E[R] =& [2pq + (N-2)q^2][1] && \text{(4.8a)}\\
&+[q^2][1] && \text{(4.8b)}\\
&+[p^2][0] && \text{(4.8c)}\\
&+[2(N-2)pq][1] && \text{(4.8d)}\\
&+[1-(2pq+(N-2)q^2)-q^2-p^2-2(N-2)pq][2] && \text{(4.8e)}\\
=& 2-2p-\frac{(1-p)^2}{N-1} && \text{(4.8f)}
\end{aligned}$$

I simplify the result and we can see that $E[R]$ has approximately linear relationship to $p$ as $N$ is large.

## 4.3 Comparison Between Two Caching Methods

Figure 4.1 shows $E[R]$ versus $p$ by the two proposed caching methods when $N = 2, 10, 50, 100$. Recall that $p$ is set to be the largest, so the red line indicates the lower bound $1/N$ of $p$ and we focus on the range right to it.

Among these four cases, "proportional to preference caching" has lower $E[R]$ in a small range starting from $p = 1/N$. However, as $p$ is increasing, "complete files caching" outperforms "proportional to preference caching" over the following much larger range. The small range where "proportional to preference caching" performs better shrinks as $N$ is increased. Furthermore, "proportional to preference caching" causes only little more traffic reduction than "complete files caching" in these small $p$ ranges. In my considered particular preference distribution for 2-user case, "complete files caching" is the more effective way to reduce traffic.

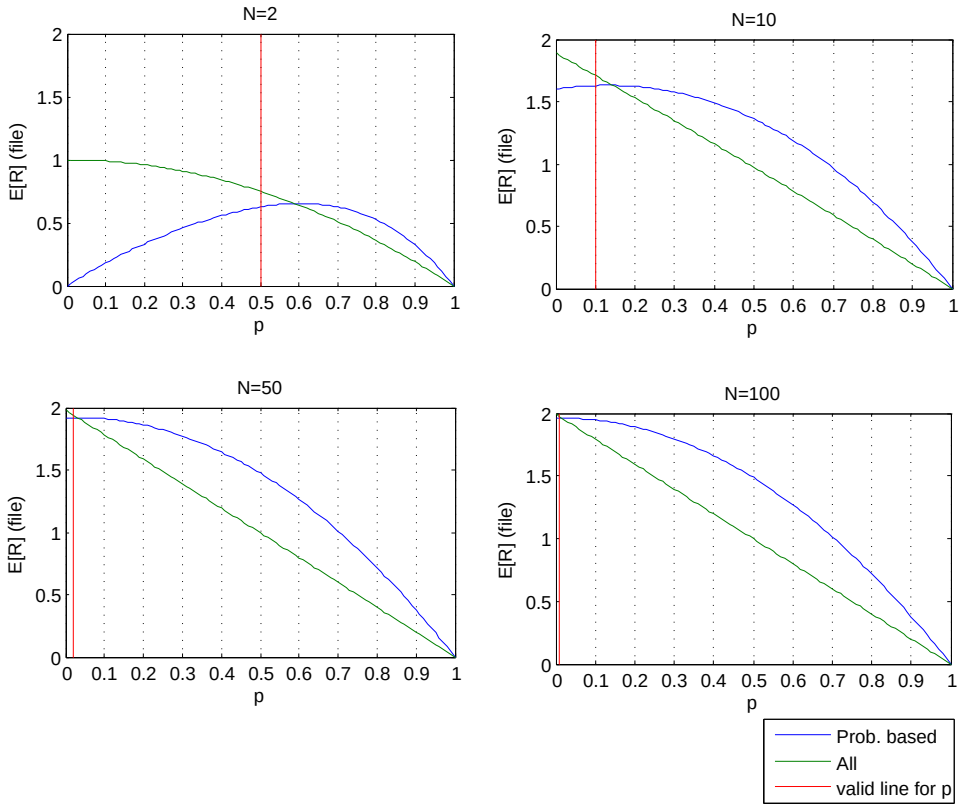



Figure 4.1: 2-user: comparison between "proportional to preference caching" and"complete files caching" when $N = 2, 10, 50, 100, M = 1$

## 4.4 General Caching

To develop a formula so that we can let MATLAB calculate different access distributions or caching methods, I let $N_i$ be the file index that user $i$ requests. To illustrate, in case the assumed access distribution and random caching are used, if user 1 requests file 2, $N_2 = 1$, so $N_i$ is a random variable with outcome space= $\{1, 2, \dots N\}$ with the user $i$'s preference distribution.

### Codable Length

Again, I care those bits of requested files that are not in the caches. First I examine the codable length for two users respectively. For user $i$,"codable length" means the bits of its requested file $n$ that are not in its cache but in the other's. Since codable length of each user is different, remaining bits may exist and need sending. In this step, the server needs to send the maximum codable length.

For only two users in the system, the codable length of user 1 is $(1 - C_{1,N_1})C_{2,N_1}$, while the codable length of user 2 is $C_{1,N_2}(1 - C_{2,N_2})$. Hence, the server sends the coded bit length equal to

$$R_{codable} = \max\{(1 - C_{1,N_1})C_{2,N_1}, C_{1,N_2}(1 - C_{2,N_2})\} \tag{4.9}$$

### Uncodable Length

The uncodable length means those bits neither of user has. User 1 has uncodable length $(1-C_{1,N_1})(1-C_{2,N_1})$, while user 2 has $(1 - C_{1,N_2})(1 - C_{2,N_2})$. There will be two cases, depending on whether two users request the same file. If they happen to request the same one, the server only sends the uncodable length (same to both users) once since the consider bits are the same. However, the server needs to send both the uncodable lengths. For convenience, I define $\delta(N_1, N_2, \dots, N_k)$ as follows,

$$\delta(N_1, N_2, \dots, N_k) = \begin{cases} 1 & \text{, if } N_1 = N_2 = \dots = N_k \\ 0 & \text{, otherwise} \end{cases}$$

The server sends the uncodable length

$$R_{uncodable} = \sum_i (1 - C_{1,N_i})(1 - C_{2,N_i}) - \delta(N_1, N_2)(1 - C_{1,N_1})(1 - C_{2,N_1}) \tag{4.10}$$

Total transmission rate is

$$R = R_{codable} + R_{uncodable} \tag{4.11}$$

### 4.4.1 Find Minimum $E[R]$ in General Caching in 2-user Case

I aim to find a caching policy $f : \mathbf{A} \to \mathbf{C}$. I continue using the assumed case: user $k$ has the largest probability of accessing the $k$ file with probability $p$ and uniform over the others. I additionally let $N = K$. For two users, the access array is

$$\mathbf{A} = \begin{pmatrix} p & 1-p \\ 1-p & p \end{pmatrix}$$

In the following, I apply the more limited policy: a one-to-one mapping $g : A_{ij} \to C_{ij}$, that is, caching fraction is uniquely depending on preference probability. To relieve notation, let $\alpha = g(p)$, and the caching array is

$$\mathbf{C} = \begin{pmatrix} \alpha & 1-\alpha \\ 1-\alpha & \alpha \end{pmatrix}$$

Then the expected transmission

$$\begin{aligned} E[R(p,\alpha)] =& E[R_{codable} + R_{uncodable}] && \text{(4.12a)} \\ =& 2p(1-p)\big[\max\{(1-\alpha)^2, \alpha^2\} + (1-\alpha)\alpha\big] && \text{(4.12b)} \\ &+ p^2\big[\max\{(1-\alpha)^2, (1-\alpha)^2\} + 2(1-\alpha)\alpha\big] && \text{(4.12c)} \\ &+ (1-p)^2\big[\max\{\alpha^2, \alpha^2\} + 2(1-\alpha)\alpha\big] && \text{(4.12d)} \end{aligned}$$

After simplification, it can be shown

$$E[R(p,\alpha)] = \begin{cases} -(2p^2 - 2p + 1)\alpha^2 + (4p^2 - 6p + 2)\alpha + (-p^2 + 2p) & , 0 \le \alpha \le 0.5 \\ -(2p^2 - 2p + 1)\alpha^2 + (-2p + 2)\alpha + (p^2) & , 0.5 \le \alpha \le 1 \end{cases}$$

For each $p \in [0, 1]$, we aim to find the minimum of $E[R(p,\alpha)]$ and the corresponding $\alpha$. Since $2p^2 - 2p + 1 = p^2 + (1-p)^2 > 0$, $E[R(p,\alpha)]$ is composed of two downward parabolas connecting at $\alpha = 0.5$, I compare the local minimums at end points $0, 0.5, 1$ and then the global minimum of $E[R(p,\alpha)]$.

A quick method is to see which side the $V(p,\alpha)$ lies to the middle points of intervals [0,0.5) and [0.5,1], or say 0.25 and 0.75 respectively.

$$V(p,\alpha) = \begin{cases} \dfrac{2p^2 - 3p + 1}{2p^2 - 2p + 1} = \dfrac{(2p-1)(p-1)}{2p^2 - 2p + 1} & , 0 \le \alpha \le 0.5 \\ \dfrac{1-p}{2p^2 - 2p + 1} & , 0.5 \le \alpha \le 1 \end{cases}$$

For example, on $0 \le \alpha \le 0.5$, minimum in this interval happens at $\alpha = 0.5$ if $V(p,\alpha)$ for this interval less than 0.25, while happens at $\alpha = 0$ if the vertex for this interval greater than 0.25. After knowing where the minimum happens in the two interval, we can infer which one has the global minimum by using the fact that the two parabolas connect at $\alpha = 0.5$. In this analysis, the situation where the minimums occur at 0 and 1 (we have to calculate their corresponding minimums so that the global one can be determined) does not happen. By this method, we can quickly know the $\alpha$ making the minimum $E[R(p,\alpha)]$.

I plot $E[R(p,\alpha)]$ versus $\alpha$ in cases of $p = 0, 0.1, \ldots, 1$ in Figure 4.2. The minimums happen at 0 when $p = 0, 0.1, \ldots 0.3$, at 0.5 when $p = 0.4, 0.5, 0.6$, while at 1 when $p > 0.6$. Here I only plot a few curves. To determine the exact turning points, we have to shorten the interval of $p$.

Figure 4.3 shows not only the minimum $E[R]$ curve I find in previous analysis but also the curves of "proportional to preference caching" and "complete files caching." The bottom plot tells the $\alpha$ corresponding to the minimum $E[R(p,\alpha)]$ for every $p$.

Actually my considered access array is symmetric so we can see the least $E[R]$ curve is symmetric to 0.5. Although the previous mentioned two caching methods work for $p \geq 0.5$, we can mirror them about the purple line because of symmetry. The weight $\alpha$ is discretized since we perform **max** operation on $\alpha$ and $1-\alpha$ and the parabola in each interval is downward. In addition, the turning points 0.4 and 0.6 match my previous analysis but here I still use a relatively long interval of $p$, 0.01, and need to set smaller interval length to derive more precise value.

The caching policy is hence interesting: a user caches all the file whose preference probability is higher than the threshold, while equally caches if both preference probabilities do not excess the threshold.

## 4.5 General Caching in Three-user Case

Now let's move into the 3-user case.

### Codable Length

It's more complicated in 3-user case and I categorize the codable lengths into two. One is that given one user, all the other users have the wanted bits but itself doesn't. The other is that exactly one of other users has the required bits.

In the first category, the server **XOR** these three codable lengths:$(1-C_{1,N_1})C_{2,N_1}C_{3,N_1}$, $C_{1,N_2}(1-C_{2,N_2})C_{3,N_2}$, $C_{1,N_3}C_{2,N_3}(1-C_{3,N_3})$. The server directly sends the remaining bits due to others' insufficient codable lengths. The result is, again, the maximum of these three codable lengths.

$$R_{codable1} = \max\{(1-C_{1,N_1})C_{2,N_1}C_{3,N_1}, C_{1,N_2}(1-C_{2,N_2})C_{3,N_2}, C_{1,N_3}C_{2,N_3}(1-C_{3,N_3})\} \tag{4.13}$$

For the second category, the server **XOR**s codable lengths for each pair of users. To illustrate, let's see how the server **XOR**s the codable lengths of user 1 and 2. Because each user must be able to get the requested file by **XOR**ing, the server won't consider the codable length neither of them has. The considered codable lengths are $(1-C_{1,N_1})C_{2,N_1}(1-C_{3,N_1})$, $C_{1,N_2}(1-C_{2,N_2})(1-C_{3,N_2})$. The server needs to send the larger of these two

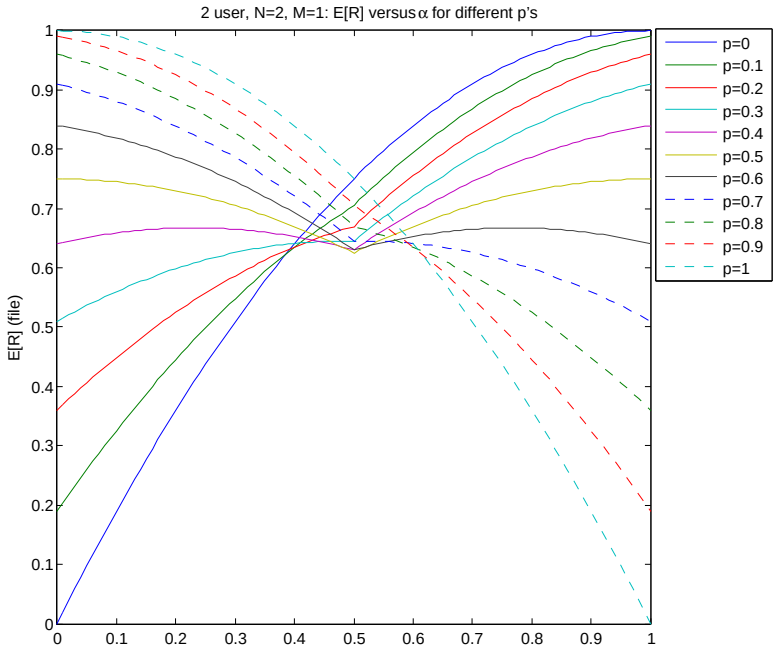



Figure 4.2: 2-user: $E[R]$ versus $\alpha$ for $p = 0, 0.1, \ldots, 1$, $N = 2, M = 1$

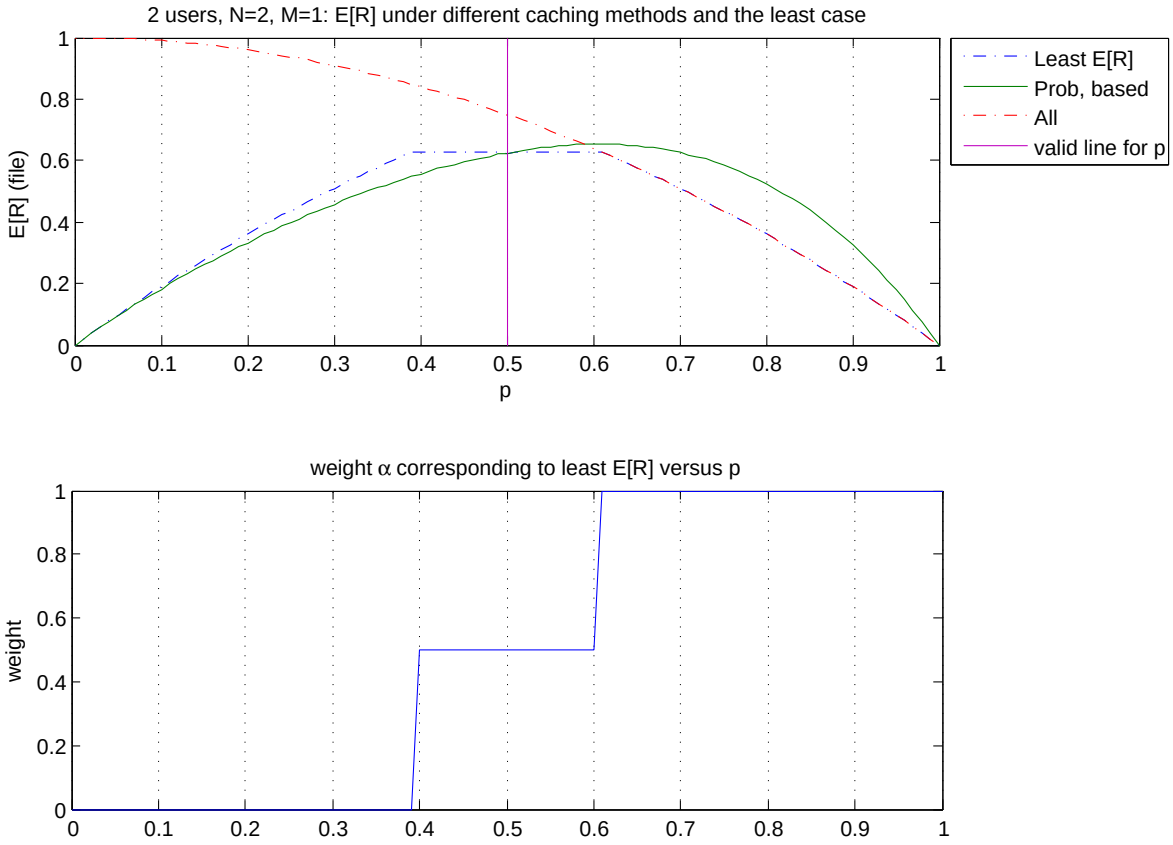



Figure 4.3: 2-user: comparison of least $E[R]$ and caching weight $\alpha$ versus $p$, $N = 2, M = 1$

like the 2-user case. Similarly, the server sends

$$R_{codable2} = \max\{(1-C_{1,N_1})C_{2,N_1}(1-C_{3,N_1}), C_{1,N_2}(1-C_{2,N_2})(1-C_{3,N_2})\} \tag{4.14a}$$
$$+ \max\{(1-C_{1,N_2})(1-C_{2,N_2})C_{3,N_2}, (1-C_{1,N_3})C_{2,N_3}(1-C_{3,N_3})\} \tag{4.14b}$$
$$+ \max\{C_{1,N_3}(1-C_{2,N_3})(1-C_{3,N_3}), (1-C_{1,N_1})(1-C_{2,N_1})C_{3,N_1}\} \tag{4.14c}$$

**Uncodable Length**

If any two of them request the same file, the server can reduce one uncodable length. The result is like "principle of inclusion and exclusion"—sum all uncodable lengths for each user, then consider if any two of them request the same file, and finally add back the term in case all the user request the same. The result is

$$R_{uncodable} = \sum_i (1-C_{1,N_i})(1-C_{2,N_i})(1-C_{3,N_i}) \tag{4.15a}$$
$$- \sum_i \delta(N_i, N_{i+1})(1-C_{1,N_i})(1-C_{2,N_i})(1-C_{3,N_i}) \tag{4.15b}$$
$$+ \delta(N_1, N_2, N_3)(1-C_{1,N_1})(1-C_{2,N_1})(1-C_{3,N_1}) \tag{4.15c}$$

where the number is represented in 3-modulo.

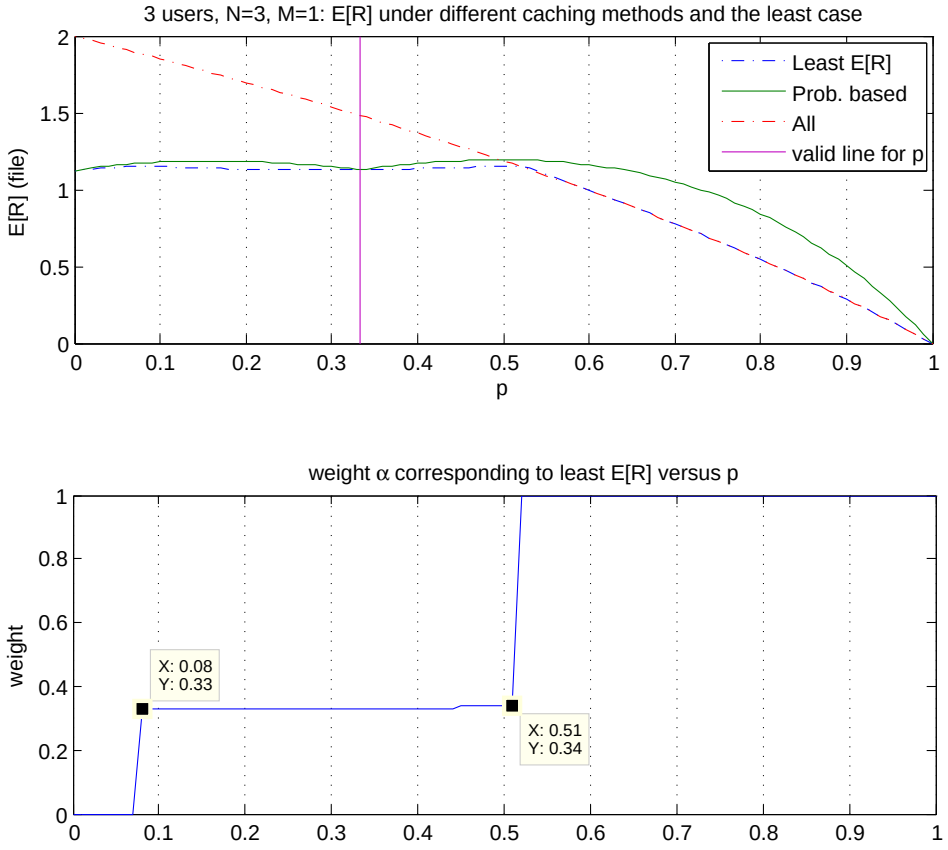



Figure 4.4: 3-user: comparison of least $E[R]$ and caching weight $\alpha$ versus $p$, $N = 3, M = 1$

### 4.5.1 Find Minimum $E[R]$ in General Caching in 3-user Case

Following my assumption in section 4.4, the considered access array for 3-user is

$$\mathbf{A} = \begin{pmatrix} p & \frac{1-p}{2} & \frac{1-p}{2} \\ \frac{1-p}{2} & p & \frac{1-p}{2} \\ \frac{1-p}{2} & \frac{1-p}{2} & p \end{pmatrix}$$

and the caching array becomes ($\alpha = g(p)$)

$$\mathbf{C} = \begin{pmatrix} \alpha & \frac{1-\alpha}{2} & \frac{1-\alpha}{2} \\ \frac{1-\alpha}{2} & \alpha & \frac{1-\alpha}{2} \\ \frac{1-\alpha}{2} & \frac{1-\alpha}{2} & \alpha \end{pmatrix}$$

Figure 4.4 shows the minimum $E[R]$, "proportional to preference caching", and "complete files caching" for 3-user case. The bottom plot tells the $\alpha$ corresponding to the minimum $E[R(p, \alpha)]$ for each $p$.

There's no symmetry in 3-user case. When $p$ excesses about 0.51, "complete file caching" is favorable. Not caching $p$ file but other two performs well if $p < 0.08$. Among the remaining range of $p$, the user caches equally the 3 files as if they seems indifferent in probability.

The weight $\alpha$ is again discretized since we perform **max** operation on $\alpha$ and $(1 - \alpha)/2$. I have not yet analyzed how the $E[R]$ versus $\alpha, \forall p \in [0, 1]$, but we still can know it's a polynomial with the largest degree equal to 3 in terms of $\alpha$.

We can see that for $p > 0.51$, caching complete files can effectively reduce $E[R]$, while the least $E[R]$ remains almost the same for $p \leq 0.51$ even if we change caching method. However, remember it is the result we choose the best caching policy; things will be worse if we don't apply different caching methods.

The caching policy is still surprising: a user caches the $p$ file once $p$ is higher than a threshold, while equally caches all the files if $p$ is not high enough but still has influence. If $p$ is too low, the user will not cache it at all but the other two equally. It is exactly what we observe in 2-user case.

# Chapter 5

# Conclusion

I give a more comprehensible proof in an Index Coding problem, relying on the concepts of Information Theory and Graph Theory. In many special symmetric graphs, we find that XOR coding cycle is good enough to achieve the outer bound on sum rate. For the case of unequal messages, we also use linear coding to reach the region derived by Theorem 2.1.

With an additional placement phase, the server can actively design the caching scheme to reduce the load in the delivery phase. I extend the cut-set lower bound in [6] to two groups with different cache memory size, and even to multiple $G$ groups. I compare the margins of decentralized and centralized algorithms. The result show that simply applying centralized schemes within each group leads to margins bounded by a constant that varies, however, with the amount of total groups since the coding opportunity between groups is not exploited.

I analyze the basic two user case with non-uniform but symmetric demands. Two intuitive caching schemes are investigated and then generalized to find an optimal expected rate by MATLAB. A surprising finding reveals that the optimal cache memory allocation is discretized due to the **max** operation in coding. This phenomenon occurs in three user case too. It suggests that we can group files based on their probability range, allocate the equal size of memory to each file in the same group but different from those in the other groups. In other words, we may reach optimal rates by transforming the non-uniform demands into uniform ones.

Our ultimate goal is to understand the outer and inner bounds for a multimedia systems where each user has its own preference distribution. To make matters simple, we would estimate such general systems into special ones. Here we investigate the scenarios where users are equipped with different size of cache and non-uniform but symmetric demands over few files of few users . From the results, coding only with each group and discretization in allocating caching memory are helpful to create a well-performed system.

# Bibliography


[1] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Inform. Theory*, vol. 49. no. 2, Feb. 2003, pp 371-381

[2] C. Fragouli, J.-Y. Le Boudec, J. Widmer, "Network coding: An instant primer," *ACIM SIGCOMM Computer Communication Review*, vol. 36. no. 1, Jan. 2006, pp 63-68.

[3] Z. Bar-Yossef, Y. Birk, T. S. Jayram, and T. Kol,"Index coding with side information," *IEEE Trans. Inform. Theory* , vol. 57, no. 3, pp. 14791494, Mar. 2011

[4] F. Arbabjolfaei, B. Bandemer, Y.-H. Kim, and E. Sasoglu, L. Wang, "On the capacity region for index coding," *2013 IEEE International Symposium on Information Theory Proceedings (ISIT)*, pp.962–966, 2013

[5] H. Maleki, V. Cadambe, and S. Jafar, "Index coding: An interference alignment perspective," 2012 [Online]. Available: arXiv:1205.1483

[6] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching, " *arXiv:1209.5807 [cs.IT]*, Sep. 2012.

[7] U. Niesen and M. A. Maddah-Ali, "Decentralized coded caching attains order-optimal memory-rate tradeoff," *IEEE/ACM Transactions on Networking*, March. 2014.

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