Bachelor of Science in Computer Science & Engineering

Decentralized Coded Caching for The Internet of Things(IoT) Using User Cooperation

by

Tasnimatul Jannah

ID: 1504069

Department of Computer Science & Engineering

Chittagong University of Engineering & Technology (CUET)

Chattogram-4349, Bangladesh.

May, 2021

Decentralized Coded Caching for The Internet of Things(IoT) Using User Cooperation

Submitted in partial fulfilment of the requirements for
Degree of Bachelor of Science
in Computer Science & Engineering

by

Tasnimatul Jannah
ID: 1504069

Supervised by

Dr. Asaduzzaman
Professor
Department of Computer Science & Engineering

Chittagong University of Engineering & Technology (CUET)
Chattogram-4349, Bangladesh.

The thesis titled '**Decentralized Coded Caching for The Internet of Things(IoT) Using User Cooperation '** submitted by ID: 1504069, Session 2019-2020 has been accepted as satisfactory in fulfilment of the requirement for the degree of Bachelor of Science in Computer Science & Engineering to be awarded by the Chittagong University of Engineering & Technology (CUET).

Board of Examiners

Declaration of Originality

This is to certify that I am the sole author of this thesis and that neither any part of this thesis nor the whole of the thesis has been submitted for a degree to any other institution.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. I am also aware that if any infringement of anyone's copyright is found, whether intentional or otherwise, I may be subject to legal and disciplinary action determined by Dept. of CSE, CUET.

I hereby assign every rights in the copyright of this thesis work to Dept. of CSE, CUET, who shall be the owner of the copyright of this work and any reproduction or use in any form or by any means whatsoever is prohibited without the consent of Dept. of CSE, CUET.

Signature of the candidate

Date:

Acknowledgements

The journey of writing this thesis involved a great deal of support, assistance and supervision from a large number of individuals, and I consider myself incredibly fortunate to have received this constant supervision and support.

Above all, I would like to thank Almighty Allah for giving me the strength to finish this work.

I would like to express my heartfelt gratitude to my honorable project supervisor Dr. Asaduzzaman, Professor, Department of Computer Science and Engineering, Chittagong University of Engineering and Technology, for his help in making this thesis a reality. I am grateful for his constant guidance, encouragement, and constructive suggestions, which have pushed me to think outside the box and given me the confidence to finish this thesis. I would like to extend my gratitude to Dr. Md. Mokammel Haque, Head & Professor, Department of Computer Science and Engineering, Chittagong University of Engineering and Technology for his kind advice. I also convey special thanks to all of my teachers for their valuable guidance during my four years of education. I'd like to thank my classmates, seniors, juniors, and staffs of the department for their helpful suggestions and assistance, which contributed to the project's successful completion.

Abstract

The Internet of Things (IoT) connects embedded devices to the internet to enable them to collect and share data with one another. It allows devices to communicate and cooperate with each other. These devices include anything from simple household items to sophisticated industrial tools. Collected data in IoT is extremely broad and diverse in nature however the majority of data sent over the network are cacheable. So caching can play a vital role to improve the performance of the IoT network. In caching process, copies of files are kept in a temporary storage location to make them available locally. There are some distinct ways of caching. To find the optimal approach for the IoT network is of great importance. The existing approaches are not suitable for an IoT network. If conventional approaches are applied directly without any modification, the IoT network will lose some fundamental properties of its own such as, user cooperation, decentralization and data aggregation. Hence, redesign of existing caching techniques are required to apply them on IoT network. Our proposed work has replaced the conventional uncoded caching approach with a decentralized caching approach in the IoT network. Existing decentralized caching approach has been modified to accommodate user cooperation. The need of a coordinating central server to exploit multicasting opportunity has been eliminated. A generalized equation has been developed to quantify the performance of the proposed caching scheme. This adaptation reduces the delivery rate by a minimum factor of 1.5 compared to the uncoded approach of the IoT network. Thus the proposed work offers a good caching scheme to the existing IoT network.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

The Internet of Things (IoT) is a latest communication paradigm in which devices that are connected together are able to communicate with each other [1]. The IoT network consists of devices with various capability of storage. The main constraint in designing an IoT network is the size of devices memory. In this purpose caching is a effective technique to mitigate this difficulties. Caching, also known as prefetching, is a network load reduction strategy that involves storing a portion of the content to be transmitted at user nodes cache [2]. As IoT networks is a network of devices connected to each other, cooperation in caching is highly expected. Cooperation between nodes improves the content delivery rate and most importantly reduces the workload of BS(Base Station), and network congestion of bottle-neck link between BS and the other nodes. Thus co-operative caching can improve caching efficiency of an IoT network.

1.2 Memory-rate tradeoff,R(M)

In any cache aided network, during the delivery phase, a memory-rate pair (M,R) is said to be achievable if it is achievable for all possible requests d1,...,dK [3]. We can simply say that rate of delivery phase in any caching approach is the rate over shared bottleneck link and is a function of cache memory size M. If each user can recover the requested file from the received packet from server using the cached contents, a memory rate pair (M, R) is said to be achievable [4]. If any memory rate pair (M,R) is achievable then rate R as a function of M is called the memory rate trade-off, R(M).

1.3 Local caching gain

Local caching gain is derived by allowing popular content to be available locally. If any users request for any content can be satisfied by serving it from its cache we can refer that the local caching gain. This gain is found significant when the local cache is big enough to store locally a significant portion of the total content [3].

1.4 Global caching gain

Global caching comes from enhancing both the caching and delivery phases simultaneously, ensuring that multiple demands can be met in the delivery process with a single multicast transmission. Since content placement is done without knowing what the actual demands are, it must be carefully planned such that multicasting possibilities are generated simultaneously for all potential requests during the delivery phase, if the total size of the global cache is high enough in comparison to the complete sum of content, global caching gain is significant. As a result, even though the caches are unable to cooperate, the number of the cache sizes becomes a critical device parameter [3].

1.5 Motivation

Main motivation of my work is to design an efficient IoT network where multicasting opportunities have been introduced and user nodes can enjoy each other's cooperation with an improved delivery rate. Below is a list of points that specify the motivation for the thesis work.

- In Existing IoT network, uncoded caching has been used, which can not exploit multicasting opportunity .
- Existing IoT network is dependent on a central coordination system.
- Decentralized coded caching is good caching approach but existing one is not suitable for IoT network.

1.6 Objective

- To identify the shortcoming of existing IoT network regarding caching scheme and user cooperation.
- To replace the uncoded caching approach of the existing IoT network with a better coded caching scheme.
- To modify selected caching scheme and make it suitable for the IoT network.

1.7 Contribution of the thesis

- Existing decentralized coded caching algorithm has been modified. In modified algorithm, user cooperation has been introduced to make the caching approach adaptable to an IoT network. It makes the network less dependent on a central coordination system and improves the delivery rate.
- Memory-rate tradeoff of proposed network has been quantified by developing a general equation .
- Performance of proposed coded caching approach has been evaluated.

1.8 Thesis Organization

The rest of the thesis is organized as follows

1. Chapter 2 contains some terminologies related to our thesis and a brief discussion on existing caching approaches and their drawbacks.
2. Chapter 3 describes our proposed caching approach and development of the generalized equation of the proposed scheme.
3. Chapter 4 provides the performance evaluation of our proposed scheme and related discussions.
4. Chapter 5 contains a summary of the contribution to the thesis and future work recommendations.

1.9 Chapter Summary

In this chapter we have provided a brief discussion on IoT network and caching. We have also discussed some related terms here to get a clear idea. After that we have clearly stated the motivation behind this work. Finally we have described the contributions to the thesis work in a precise manner.

Chapter 2

Literature Review

2.1 Introduction

The focus of this thesis is to design an efficient IoT network with an improved delivery rate by introducing a new caching approach. This chapter discusses about various caching approaches from previous works and their limitations. It also discusses the existing IoT network and its shortcomings in a brief.

2.2 Internet of Things (IoT)

The Internet of Things (IoT) relies on communication, caching, and computation in wireless networks to link a wide range of devices [1, 2]. It is a concept in which everyday objects can be fitted with detecting, sensing, networking, and processing capacities, allowing them to communicate with one another, seek for neighbours cooperation as well as other devices and services over the Internet to achieve a goal [5, 6]. Application field of IoT is so vast that it can be divided into two major categories. In massive IoT application, low cost and low power consuming devices are used to create the network. On the other hand critical IoT applications are high demanding in sense of reliability and availability. From this categorization it is seen that IoT technology is growing tremendously and thus leaving significant amount of burden on designing the wireless network [1].

2.3 Overview of Caching

Caching is a method of reducing traffic rates which prefetches contents in the end users' memories [3]. Caching makes use of network-wide memories to duplicate content during off-peak periods. Those are used in requested content delivery with less strain on the network during peak traffic hours [7]. Any caching-aided network has two phases in network operation, content placement and content delivery phase.

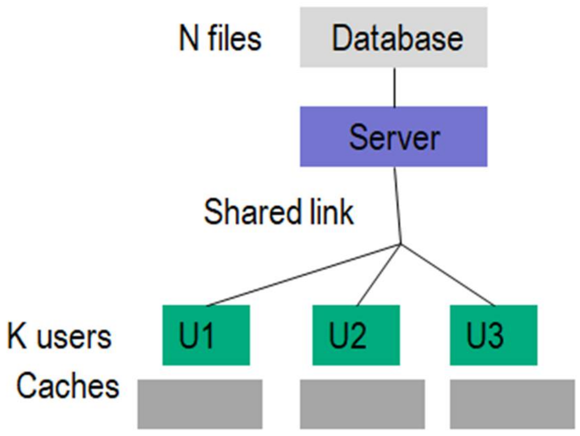


Figure 2.1: Cache Network

- **Content placement phase**- In content placement phase, users have direct access to database to fill up their caches [3]. Server pre-allocates some of the data in cache memories when network is in off-peak hours. Placement phase is constrained by size of cache size of users [8].
- **Content delivery phase**- In content delivery phase, only server has the access to files in database [3]. Requested contents are being delivered by server during peak hours. User reconstruct requested file from local cache and transmitted signal. Delivery phase is constrained by the delivery rate [8]. The main goal is to design a cache approach to minimize the delivery rate during the delivery phase.

Caching can be both uncoded and coded. These are described in next sections.

2.3.1 Uncoded caching

It's also known as conventional scheme of caching. Main idea behind this is to deliver partial content locally from caches that are situated nearby [7]. [3] introduces uncoded caching scheme. It is an approach where neither the placement process nor the distribution phase uses coding. In this solution, first M/N fraction of each files bits has been cached during placement phase when network resource are available and traffic is low. Thus each user has some fractions of each file. During delivery phase, server transmits remaining fraction over the shared connection. So any user can get its desired file from local cache and server. If we consider worst case scenario, we can get the delivery rate,

$$R_U(M) = K \times (1 - M/N) \tag{2.1}$$

[8] Let consider a network where ,

No. of users, K=2

No. of files, N=2

Cache size, M=1

- **Placement phase-** At placement phase, each user cacheS the first parts of the files. As this parts are of F/2 size, so they will satisfy memory constraint.

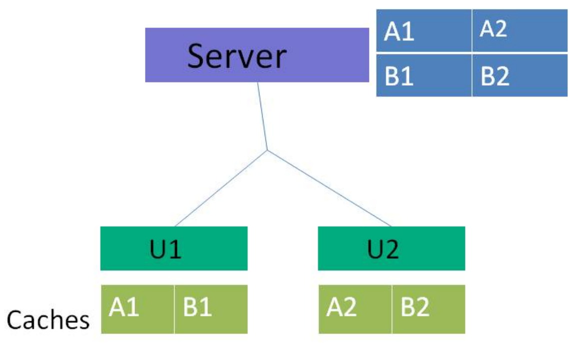


Figure 2.2: Placement phase of uncoded caching

- **Delivery phase-** At delivery phase we have considered the worst case scenario, each users demand for different files. So server has to transmit A2,B1 to user1 and user2.

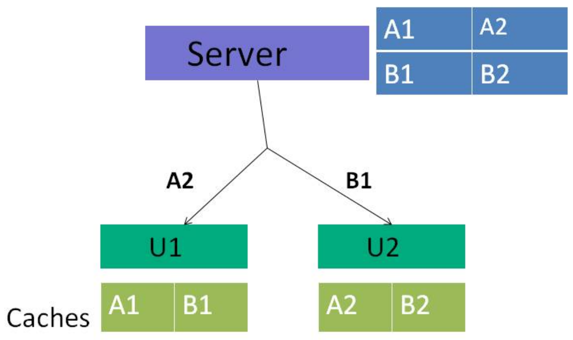


Figure 2.3: Delivery phase of uncoded caching

Thus each user can recover their requested files from local caches and transmitted message over shared link. As server has to transmit ½ of each files separately to each user so the rate will be, $R_U(M)=1$(As two distinguished transmission i.e.½ +½).

2.3.2 Coded caching

Coded caching is a caching approach where either placement phase or delivery phase uses coding. The placement and delivery phases of coded caching are designed to allow multicasting among users with varying needs [9]. There are distinct approaches used in coded caching. These are described below

2.3.2.1 Coded placement

In uncoded placement caching, the server places uncoded portions of each file in the users' cache memories [10–12]. But in systems with coded placement, the server positions files in users' caches that have been coded. This pieces of files are then decoded using the transmissions from server during the delivery phase [13, 14]. Users cache both uncoded and coded pieces of files in proposed system of [15]. Users with large memories decode coded pieces with unicast/multicast signals received from the server during delivery phase and serve users with smaller memories. We find that as the gap between the cache sizes grows, the benefit from coded placement grows, but decreases as the number of files grows. According to a recent reference [16], when the cache sizes of the two users vary, coded placement is vital to achieve delivery load which is optimal in a two-user scheme. Motivated by coded placement introduced in [15], a caching scheme is suggested for networks where multiple users share the same cache memory in [17]. A coded placement scheme has been suggested that takes advantage of the asymmetry in the number of users associated with each cache. The caches are filled with both coded and uncoded bits of the library files in this proposed scheme. Specifically, uncoded pieces are being placed in the cache that is shared by a greater number of users, therefore storing coded pieces in the remaining caches. This action depends on the network connection pattern.

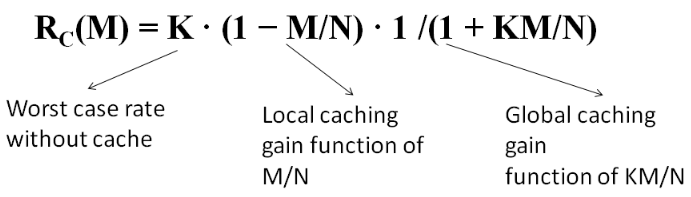
2.3.2.2 Centralized Coded Caching

In centralized coded caching scheme, central server knows the number of users and their identities. As a result, cache contents can be configured jointly in advance to optimize multicasting opportunities [9]. The caches are organized by a central server such that each subset of the cache memories shares a particular portion of the content. This carefully planned overlap among the cache memories ensures that coded-multicasting capabilities are available for all potential user demands at the same time [8]. The rate of bottleneck can be written as,

$$R_C(M) = K \times (1 - M/N) \times (1/(1 + KM/N)) \tag{2.2}$$

This rate is formed by three distinct factors.

This global gain can be viewed as a multicasting gain that is usable for all imaginable demands at the same time. When the number of users is less than the number of files, the worst-case scenario is that each user requests a different file. As a result, there are no natural opportunities for multicasting. Even among users who request different files, the scheme carefully designs content placement in order to establish coded multicasting opportunities in the delivery process [3].



$R_C(M) = K \cdot (1 - M/N) \cdot 1/(1 + KM/N)$
Worst case rate without cache
Local caching gain function of M/N
Global caching gain function of KM/N

Figure 2.4: Factors in equation of centralized coded caching

[18] Let consider a network where ,

No. of users, K=2

No. of files, N=2

Cache size, M=1

Files A, B are splitted into two subfiles. These are A=(A1, A2) and B=(B1, B2).

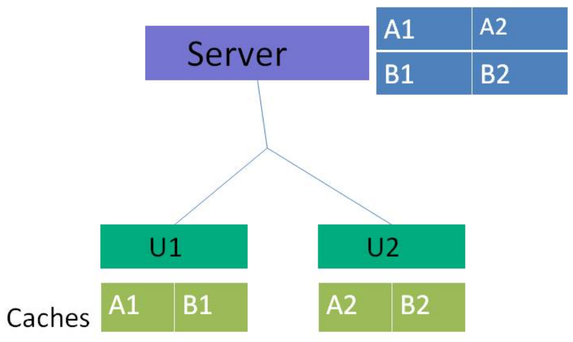


Figure 2.5: Placement phase of centralized coded caching

- **At placement phase-** User1 caches file fragment A1, B1 and user2 caches A2, B2.

- **At delivery phase-** Let, user requirement tuple is (A,B). We can see that user1 already has A1 and user2 already has B2, their required portions are available in others cache but this nodes can not exchange files.

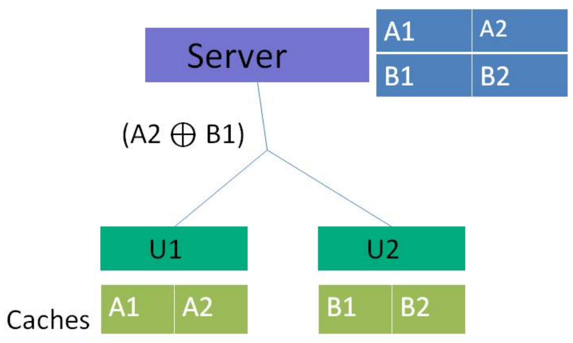


Figure 2.6: Delivery phase of centralized coded caching

By designing content placement in a particular pattern, coding has been used to reduce transmission rate. Instead of sending missing part with a rate of 1, server sends a coded signal A2 $\oplus$ B1. As server has to transmit ½ of each files in a coded manner so the rate will be

$R_C(M)=1/2$ (as two distinguished fraction of files are bit wise XORed).

Centralized approach can exploit coded-multicasting opportunity but requires a central coordinating server. We may face some situations where during delivery phase, the identity or even the number of active users in the delivery phase will not be determined for many hours. Another example is that the placement phase may be carried out in one network while the distribution phase is carried out in another. Coordination during the placement phase may not be feasible in either case. In this certain situations, a central coordinating server might not be useful, raising the question of whether this multicasting gain can still be achieved in a more decentralized setting [8].

2.3.2.3 Decentralized Coded Caching

Decentralized coded caching is an approach where in the placement phase, it can work for an unknown number of users in isolated networks acting independently of one another [8]. The most significant aspect of this placement scheme is that each cache is filled regardless of the number or identity of other users in the system [18]. [8] shows us the placement and delivery phase of this scheme. The delivery rate for decentralized coded caching can be written as

$$R_D(M) = K \times (1 - M/N) \times min\{(N/KM) \times (1 - (1 - M/N)^K), N/K\} \quad (2.3)$$

[8] Let, No. of user, K=2

No. of files, N=2

M $\in [0, 2]$

S $\subset \{1, 2\}$

- **Content placement phase-** At placement phase, each file is divided into 4 subfiles.

  A=(A0, A1, A2, A12)

  B=(B0, B1, B2, B12)

  each user caches some bits of each files.

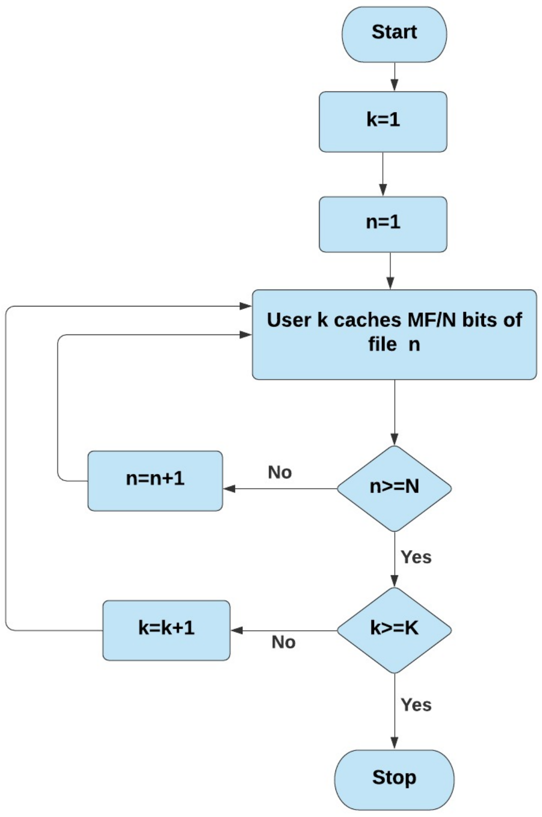
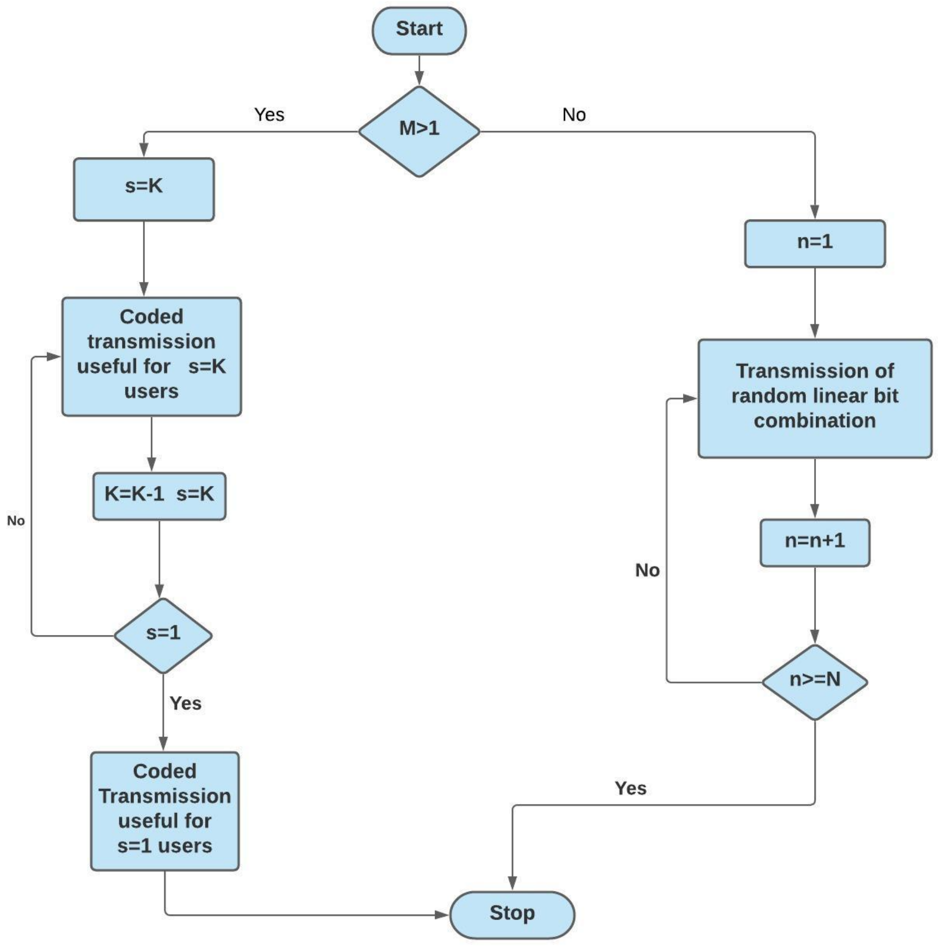


Figure 2.7: Placement phase of decentralized coded caching

- **Content delivery phase-** At content delivery phase, let requirement tuple is (A,B). User1 needs A2 which is in cache of user2 and user2 needs B1 which resides in cache of user1. But they can not exchange file. This approach starts with s=2. Server sends A2 $\oplus$ B1. It transmits information that is useful for two user simultaneously. Then at second iteration, s=1, server sends A0, B0 which are useful for only one user. As server sends ¼ file at first iteration in coded form then sends two ¼ portion of files so the rate is, R$_D$(M)=3/4



Start
M>1
Yes
No
s=K
n=1
Coded transmission useful for s=K users
Transmission of random linear bit combination
K=K-1 s=K
No
n=n+1
No
s=1
Yes
n>=N
Coded Transmission useful for s=1 users
Yes
Stop

Figure 2.8: Delivery phase of decentralized coded caching

Decentralized Coded caching can handle some issues more effectively then centralized coded caching. It can handle asynchronous demands. The server identifies the collection of active users sharing a bottleneck connection before attempting delivery. It then uses a decentralized coded caching delivery scheme for that group of users. Thus it can handle dynamic network also [18]. The popularity of content files often varies by many orders of magnitude. [19] recommends splitting the collection of files into many groups of similar popularity within each group for such nonuniform distributions. Each group is then given a portion of the memory. The decentralized coded caching scheme's placement and delivery phases are then implemented to each group of files. Thus it can handle nonuniform demands. In online caching scheme, the universality of decentralized coded caching is used because the number of requests from cached files is random. [20] shows how it exploits property of decentralized coded caching approach in online caching.

2.3.3 Co-operative caching

In IoT network, devices that link together can communicate with each other [1]. To cache and exchange content items in the IoT, objects and devices must cooperate. Usually IoT connects devices with various cache size, so the devices has capability to cache popular contents. They can help each other in term of content retrieval by providing cache content locally. So that a node can retrieve a data content from a nearby user node instead of BS. This can be defined as cooperative caching. In a conventional cooperative caching approach, popular contents are placed to cache nodes by BS at content placement phase. Due to the limited capacity, each cache node chooses a subset of contents to cache. Contents can be accessed from cache nodes rather than BS [21]. Content exchanging capacity of user nodes is reliant on historic information like social relationship, rather than link quality, energy etc [22]. User nodes can show varying degrees of altruism. Devices can refuse to share resources with strangers, but social behaviors may encourage cache and content sharing. In other words, if devices have close social links, they are more likely to share resources with others [23].

2.4 Background of the Problem

As IoT is growing so fast, designing a efficient IoT network is quite challenging, so many ideas are being proposed. [1] suggested to combine co-operative caching with social networking. This network will use computational and storage capability of user node which is idle at that moment. Social relationships, cooperative behavior of nodes are examined, with a focus on their willingness of sharing resources. The collected data is sent to the BS. Data is cached in users with unused storage capacity and similar interest. When any node requires any resource it can collect fragments from other users instead of BS and can retrieve its desired content. But here uncoded caching approach has been used. Uncoded caching can not exploit multicasting opportunity and thus it can't enjoy global caching gain.

[18] described a approach is called centralized coded caching. Centralized caching approach can exploit coded-multicasting opportunity but requires a central coordinating server.

[8] showed us multicasting gain can still be achieved in a more decentralized setting i.e a decentralized approach which gives us more flexibility and rate is closer to centralized approach. But decentralized coded caching doesn't support user cooperation. User cooperation is highly needed in any IoT network. So decentralized coded caching is not directly applicable in to any IoT network.

2.5 Present state of the problem

We have addressed the shortcomings of the current algorithm in the background section. Now, we'll discuss the solutions. First of all, we have to find out a coded caching approach to replace the uncoded caching of existing IoT network to exploit multicasting opportunity. We have compared both centralized and decentralized approach. We have seen that centralized approach has a good rate but decentralized approach has better flexibility and can operate without a central co-ordination system. Rate of decentralized coded caching is closer to centralized. So it's a better coded caching approach for IoT network. But in this approach user nodes doesn't communicate to each other i.e, it's not a cooperative caching approach. On the other hand, IoT demands high user cooperation as devices are connected to each other able to communicate and cooperate. So, our selected caching approach cannot be deployed directly. It has to be modified to be fitted into requirement of a IoT network. We have solved this conflict by modifying the existing decentralized coded caching approach. Later we have developed an equation to quantify the R(M) of our proposed algorithm.

2.6 Chapter Summery

In this section we have described various existing caching approaches. We have seen that caching can be both uncoded and coded. Coded caching has two different ways also. We have learnt about advantages of decentralized coded caching.

We have also discussed cooperative caching in a brief manner. Finally, we have pointed out the shortcomings of existing algorithms and their incompetence to be incorporated directly into an IoT network. This section has been concluded by stating present condition of the problem and the probable solution that we will provide through our proposed work.

Chapter 3

Methodology

3.1 Introduction

An IoT network has been proposed which uses decentralized coded caching in delivery phase. In this section we have discussed the whole process including the both content placement phase and content delivery phase. Then a pictorial representation of proposed algorithm in a IoT network has been shown. Later, we have shown the development of the equation of our proposed algorithm considering some different scenario and finally we have verified this new equation.

3.2 Overview of Proposed framework

3.2.1 System Setup

Our proposed approach is based on an IoT network.It is described below with appropriate details

- This network has BS that is connected with several user nodes.BS is connected with the user nodes with shared bottle neck link.
- BS has an access to database with N files.
- Each of the K user-nodes has independent cache memories of size M that can cache a certain amount of any files.
- User nodes can communicate and cooperate with each other.
- The cooperation relies on their social relationship and desire to exchange cache content.

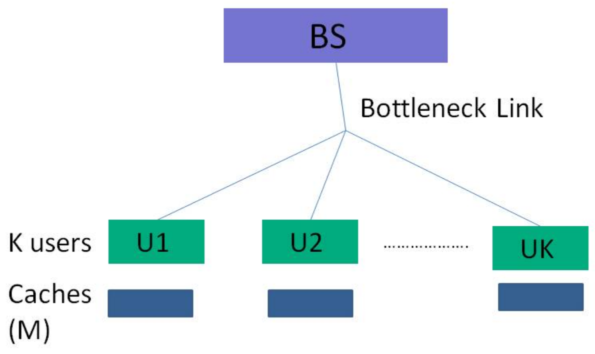


Figure 3.1: The IoT network

In figure 3.1, the IoT network has a BS(Base Station) and several user nodes U1, U2, ..., UK. User nodes are connected with the BS via a shared bottleneck link. Each user has a cache memory of size M. The users are connected to each other and are able to communicate and cooperate. Their cooperation relies on the social relationship among them. All further examples to describe our proposed scheme will be based on this network setup.

3.2.2 Proposed Algorithm

- **Content placement-** Content placement of our proposed approach is identical to the placement phase of decentralized coded caching. It starts with variable k=1 and n=1. User k will caches fragment of file n in each iteration, then value will change and thus next user can cache fragments of all n files. At first iteration of the outer loop, as k=1 that means user1 will populate its cache now. Now, iteration of the inner loop will begin with n=1 which means user1 will cache the fragments of the first file. At each iteration the value of n will be increased by 1 and user1 will fill up its cache with the fragments of corresponding file specified by the variable n. The inner loop will terminate when we will reach the value n=N, that means user1 has cached the fragments of all N files.

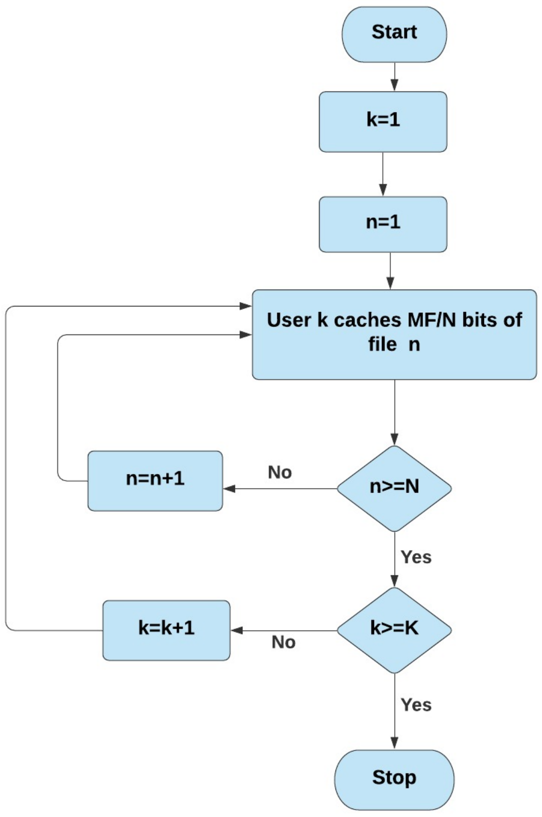


Figure 3.2: Content Placement Phase of Proposed Algorithm

Now, the outer loop will start its second iteration. The value assigned to variable k will be increased by 1 and will be k=2, that means user2 will populate its cache. The outer loop will terminate when we will reach the value k=K, that means all of the user nodes has finished caching file fragments. Placement phase is terminated when n=N and k=K is reached. It means that fragments of all files have been cached in all users cache.

- **Content delivery-** Our initial assumption is that all nodes has strong social relationship among them and they trust each other. So they can assist each other. Now, delivery phase starts with s=K, BS will transmit coded messages to satisfy more than one requests at a time. s=K means, the coded transmission will satisfy K users at once.

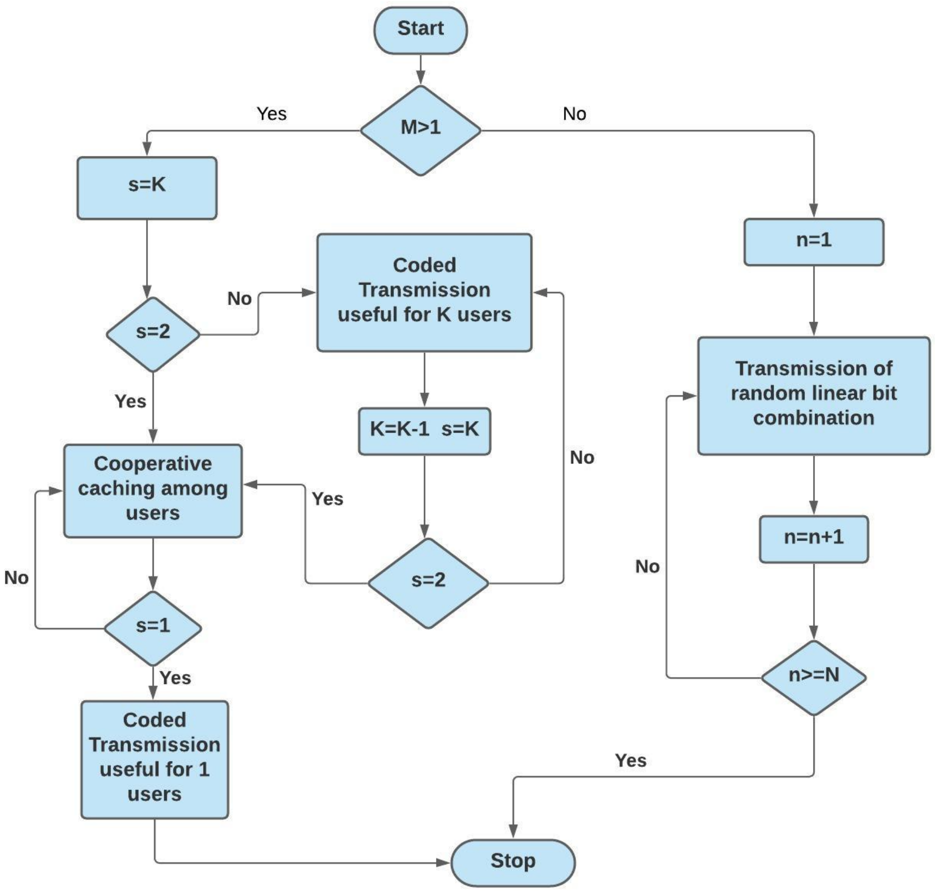

Start
M>1
Yes
No
s=K
n=1
Coded Transmission useful for K users
No
s=2
Transmission of random linear bit combination
Yes
K=K-1 s=K
No
Cooperative caching among users
Yes
n=n+1
No
No
s=2
s=1
n>=N
Yes
Coded Transmission useful for 1 users
Yes
Stop

Figure 3.3: Content Delivery Phase of Proposed Algorithm

whenever s reaches value of 2, BS wil stop sending coded messages. At that iteration user cooperation will be invoked. User will interchange file fragments to satisfy each other's requirement. When s=1, again BS will send messages that useful for only one user at a time.

3.2.3 Graphical Representation of Proposed Algorithm for N=K=2

- **Content placement**

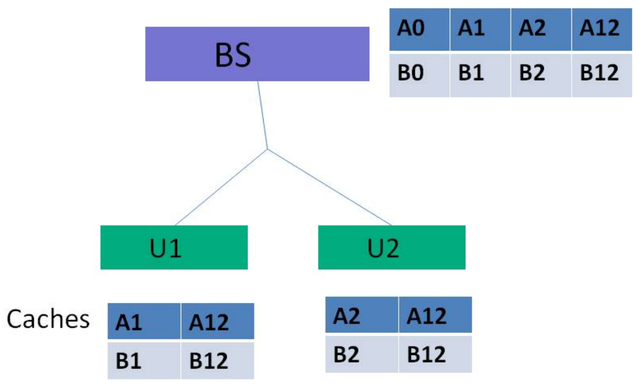


Figure 3.4: Content Placement Phase for N=K=2

At first, Files A and B are partitioned by the BS.

A=(A0, A1, A2, A12)

B=(B0, B1, B2, B12)

According to the flowchart presented in figure 3.2, iteration of outer loop will start with k=1, so user1 will fill up its cache with file fragments of file A at first iteration of inner loop when, n=1. At second iteration of inner loop, n=2, so user1 will cache the fragments of file B. The second iteration of the outer loop will begin as k=2. User2 will populate its cache with fragments of file A and then B in the same manner as user1. Cache contents of both users are shown in figure 3.4.

- **Content Delivery**

  1. When, s=2, users will attain desired file fragments via user cooperation.

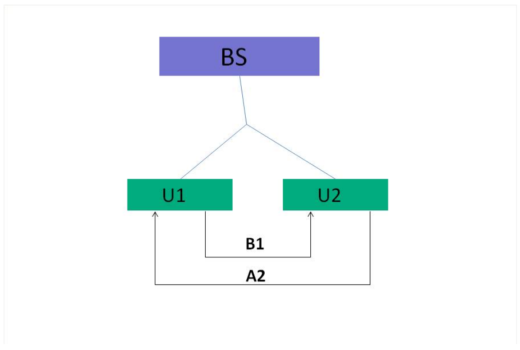


Figure 3.5: Content Delivery Phase for N=K=2 (User Cooperation at s=2)

  2. When, s=1, server will transmit file fragments useful for only one user i.e A0 and B0.

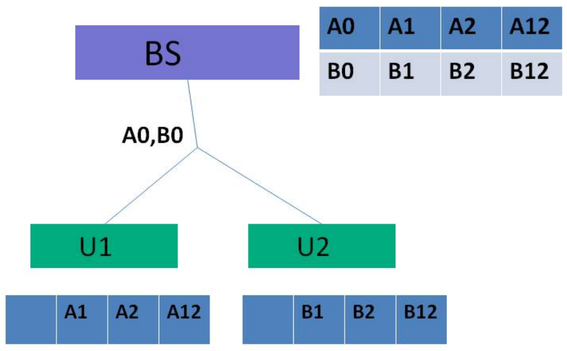
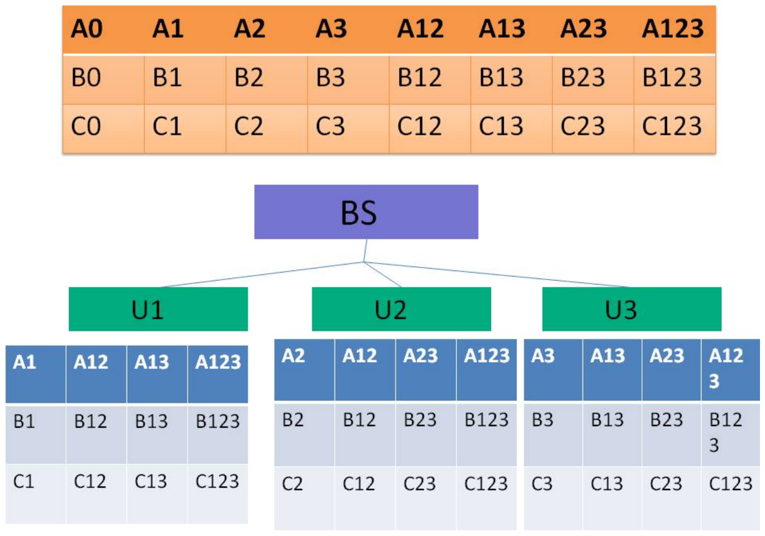


Figure 3.6: Content Delivery Phase for N=K=2(s=1)

3.2.4 Graphical Representation of Proposed Algorithm for N=K=3

- **Content placement** Content placement is shown in the figure below



A0	A1	A2	A3	A12	A13	A23	A123
B0	B1	B2	B3	B12	B13	B23	B123
C0	C1	C2	C3	C12	C13	C23	C123

U1			
A1	A12	A13	A123
B1	B12	B13	B123
C1	C12	C13	C123

U2			
A2	A12	A23	A123
B2	B12	B23	B123
C2	C12	C23	C123

U3			
A3	A13	A23	A12 3
B3	B13	B23	B12 3
C3	C13	C23	C123

Figure 3.7: Content Placement Phase for N=K=3

- **Content delivery**

  1. When s=3, server will transmit coded signal useful for 3 users. Server will multicast A23 $\oplus$ B13 $\oplus$C12.

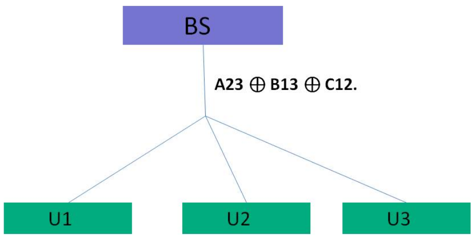


Figure 3.8: Content Delivery Phase for N=K=3 (s=3)

2. When s=2, server won't transmit any coded signal. Users will attain desired file fragments via user cooperation.

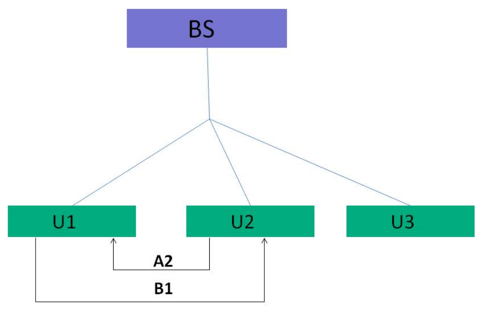


Figure 3.9: Content Delivery Phase for N=K=3 (User Cooperation at s=2)

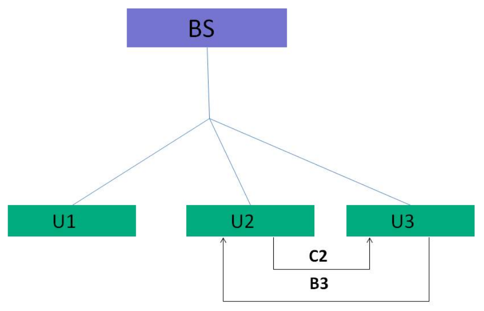


Figure 3.10: Content Delivery Phase for N=K=3 (User Cooperation at s=2)

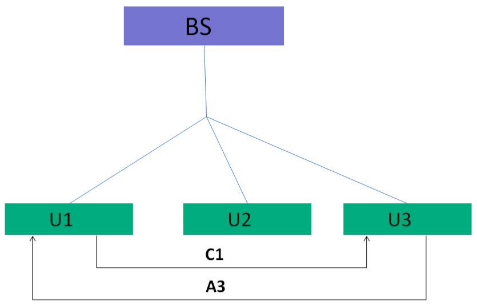


Figure 3.11: Content Delivery Phase for N=K=3 (User Cooperation at s=2)

3. When s=1, server has to send A0,B0,C0

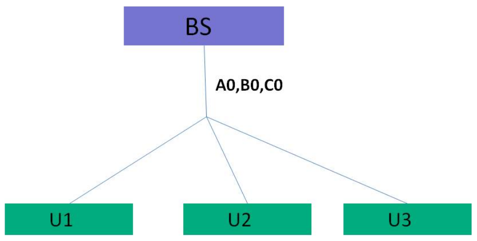


Figure 3.12: Content Delivery Phase for N=K=3 (s=1)

3.3 Reason behind choosing s=2 to invoke user cooperation

- Let consider a decentralized coded caching system where, N=K=2. Here we only considered delivery phase. Delivery phase starts with S=K, here, K=2 and we have considered worst case scenario.

  1. First iteration s=2, server will transmit coded signal useful simultaneously for 2 users. Server will multicast A2 $\oplus$ B1
  2. Second iteration s=1, server will transmit coded signal useful only for 1 users. Server will unicast A0,B0

  When s=2, it's very convenient to consider user co-operation as user can easily get required fragments. Suppose user1 needs A2 which is available at user2's cache and user2 needs B1 which is at user1's cache, so they can easily co-operate to get desired portion of file. When s=1, it's obvious that server must transmit A0, B0 as this fragments are not present in cache of users.

- Again considering a decentralized coded caching system where, N=K=3. Here we only considered delivery phase. Delivery phase starts with s=K, here, K=3 and we have considered worst case scenario.

  1. First iteration s=3, server will transmit coded signal useful simultaneously for 3 users. Server will multicast A23 $\oplus$ B13 $\oplus$ C12.
  2. Second iteration s=2, server will transmit coded signal useful simultaneously for 2 users. Server will multicast A2 $\oplus$ B1, A3 $\oplus$ C1, B3 $\oplus$ C2.
  3. Third iteration s=1, server will transmit coded signal useful only for 1 users. Server will unicast A0, B0, C0 to user1, user2, user3.

  We can see that when s=3, if we consider user cooperation, users have to decide from whom it should take the file fragments. Suppose user1 requires A23 which is available at both user2 and user3, it's time consuming and requires additional computation to determine the appropriate sender.

  When s=1, it's obvious that server must transmit A0, B0, C0 as this fragments are not present is cache of the users.

  When s=2, it's very convenient to consider user cooperation as user can easily get required fragments. Suppose user2 needs B3 which is available at user3's cache and user3 needs C2 which is at user2's cache, so they can easily co-operate to get desired portion of file.

3.4 Development of Generalized Equation

- [8] By the law of large numbers,we can write

  |AS| $\approx$ (M/2)$^{|S|}$ $\times$ (1 - M/2)$^{2-|S|}$ F as well as

  |BS| $\approx$ (M/2)$^{|S|}$ $\times$ (1 - M/2)$^{2-|S|}$ F

  This equation is suitable for the system with N=K=2. Here, S$\subset${1,2} [as we have user1 and user2]

  1. When |S|=0, |A0|/F is approximately (1 - M/2)^2. [s=1]
  2. When |S|=1, |A1|/F and |A2|/F are approximately (M/2) $\times$ (1 - M/2). [s=2]

     |B1|/F and |B2|/F are approximately (M/2)$\times$(1 - M/2). [s=2]
  3. When |S|=2, |A1, 2|/F and |B1, 2|/F are approximately (M/2)^2. But as these file fragments are already found in local caches so it won't be considered.

  When s=2, user-cooperation eliminates need of coded transmission [No transmission].

  When s=1, server only has to send A0, B0 [2 transmission].

  If we define the rate of our proposed algorithm by R$_{UC}$(M), then

  R$_{UC}$(M)=2$\times$(1 - M/2)^2.

- Now, we will consider N=K=3. By the law of large number we can write

  |AS| $\approx$ (M/3)$^{|S|}$ $\times$ (1 - M/3)$^{3-|S|}$ F

  |BS| $\approx$ (M/3)$^{|S|}$ $\times$ (1 - M/3)$^{3-|S|}$ F

  |CS| $\approx$ (M/3)$^{|S|}$ $\times$ (1 - M/3)$^{3-|S|}$ F

  Now, S $\subset$ {1, 2, 3} [as we have user1, user2 and user3]

  1. When, |S|=0, |A0|/F is approximately (1 - M/3)^3. [s=1]
  2. When, |S|=1, |A1|/F, |A2|/F and |A3|/F are approximately

$(M/3)\times (1 - M/3)^2$. [s=2]

$|B1|/F$, $|B2|/F$ and $|B3|/F$ are approximately $(M/3)\times(1 - M/3)^2$. [s=2]

$|C1|/F$, $|C2|/F$ and $|C3|/F$ are approximately $(M/3)\times(1 - M/3)^2$. [s=2]

3. When $|S|=2$, $|A1,2|/F$, $|A1,3|/F$, $|A2,3|/F$ are approximately $(M/3)^2 \times (1-M/3)$. [s=3]

   same goes for B,C

4. When $|S|=3$, $|A1,2,3|/F$ is approximately $(M/3)^3$

   same goes for B, C. But as these file fragments are already found in local caches so it won't be considered.

When s=3, server has to send $A23 \oplus B13 \oplus C12$. [1 transmission]

When s=2, user-cooperation will eliminate the need of transmission from server. [No transmission]

When s=1, server has to send A0, B0, C0. [3 transmission]

So, we can write the delivery rate as

$R_{UC}(M) = 3\times(1 - M/3)^3 + (M/3)^2 \times (1-M/3)$

- Now we will consider N=K=4. Before calculating memory rate tradeoff R$_{UC}$(M), content placement and delivery phase should be elaborated to determine number of transmissions occurred.

    1. **Content Placement**

Cache of User1	A1,A12, A13,A14, A123,A134, A124,A1234	B1,B12, B13,B14, B123,B134, B124,B1234	C1,C12, C13,C14, C123,C134, C124,C1234	D1,D12, D13,D14, D123,D134, D124,D1234
Cache of User2	A2,A12, A23,A24, A123,A234, A124,A1234	B2,B12, B23,B24, B123,B234, B124,B1234	C2,C12, C23,C24, C123,C234, C124,C1234	D2,D12, D23,D24, D123,D234, D124,D1234
Cache of User3	A3,A13 ,A23,A34, A123,A134, A234,A1234	B3,B13, B23,B34, B123,B134, B234,B1234	C3,C13, C23,C34, C123,C134, C234,C1234	D3,D13, D23,D34, D123,D134, D234,D1234
Cache of User4	A4,A14, A24,A34, A134,A234, A124,A1234	B4,B14, B24,B34, B134,B234, B124,B1234	C4,C14, C24,C34, C134,C234, C124,C1234	D4,D14, D24,D34, D134,D234, D124,D1234

Table 3.1: Content Placement for N=K=4

2. **Content Delivery-** Delivery phase starts with s=K, here, K=4 and we have considered worst case scenario.

   (a) First iteration s=4, server will transmit coded signal useful simultaneously for 4 users. Server will multicast A234$\oplus$ B134 $\oplus$ C124 $\oplus$ D123.

   (b) Second iteration s=3, server will transmit coded signal useful simultaneously for 3 users. Server will multicast A23 $\oplus$ B13 $\oplus$ C12, A24 $\oplus$ B14 $\oplus$ D12, A34 $\oplus$ C14 $\oplus$ D13, B34 $\oplus$ C24 $\oplus$ D23.

   (c) Third iteration s=2, user cooperation will elliminate need of transmission from server.

   (d) Fourth iteration s=1, server will transmit coded signal useful only for 1 users. Server will unicast A0, B0, C0, D0 to user1, user2, user3 and user4.

By the law of large number we can write,

$|AS| \approx (M/4)^{|S|} \times (1 - M/4)^{4-|S|}$ F as well as

$|BS| \approx (M/4)^{|S|} \times (1 - M/4)^{4-|S|}$ F as well as

$|CS| \approx (M/4)^{|S|} \times (1 - M/4)^{4-|S|}$ F as well as

$|DS| \approx (M/4)^{|S|} \times (1 - M/4)^{4-|S|}$ F as well as

Now, S $\subset$1,2,3,4 [as we have user1, user2, user3, user4]

* When |S|=0, |A0|/F is approximately $(1 - M/4)^4$. [s=1]

* When |S|=1, |A1|/F, |A2|/F, |A3|/F and |A4|/F are approximately $(M/4)\times$ $(1 \; M/4)^3$. [s=2]

  |B1|/F, |B2|/F, |B3|/F, |B4|/F are approximately $(M/4)\times$ $(1 \; M/4)^3$ [s=2]

  |C1|/F, |C2|/F, |C3|/F, |C4|/F are approximately $(M/4)\times$ $(1 \; M/4)^3$ [s=2]

|D1|/F, |D2|/F, |D3|/F, |D4|/F are approximately (M/4)$\times$ (1 M/4)^3 [s=2]

- When |S|=2, |A1,2|/F, |A1,3|/F, |A1,4|/F, |A2,3|/F , |A2,4|/F, |A3,4|/F are approximately (M/4)^2 $\times$(1-M/4)^2. [s=3]

  same goes for B, C, D.

- When |S|=3, |A1,2,3|/F, |A1,3,4|/F, |A2,3,4|/F, |A1,2,4|/F is approximately (M/4)^3 $\times$ (1-M/4). [s=4]

  same goes for B, C, D.

- When |S|=4, |A1,2,3,4|/F is approximately (M/4)^4.

  same goes for B, C, D.

When s=4, server will send A234 $\oplus$ B134 $\oplus$ C124 $\oplus$ D123. [1 transmission]

When s=3, server will send A23 $\oplus$ B13 $\oplus$ C12, A24 $\oplus$ B14 $\oplus$ D12, A34 $\oplus$ C14 $\oplus$ D13,B34 $\oplus$ C24 $\oplus$ D23. [4 transmission]

When s=2, user cooperation will elliminate need of transmission from server.

When s=1, server will send A0, B0, C0, D0. [4 transmission]

So, the delivery rate is R$_{UC}$(M

R$_{UC}$(M)= 4$\times$ (1 - M/3)^4 + 4 $\times$ [(M/4)^2 $\times$ (1-M/4)^2] + (M/4)^3 $\times$ (1-M/4)

Now from above calculation, we can write a generalised equation for our proposed Algorithm-

$$R_{UC}(M) = K \times (1 - M/N)^K + \sum_{i=2}^{K-1} \begin{pmatrix} K \\ i+1 \end{pmatrix} \times (M/K)^i \times (1 - M/N)^{K-i} \quad (3.1)$$

Where, K= No. of users

N= No. of files

M= Cache memory size

i= |S|

3.5 Verification of Generalized Equation

Let consider a network with N=K=5

- **Content Placement**

1. **Cache content of user1**

Fragments of file A	Fragments of file B	Fragments of file C	Fragments of file D	Fragments of file E
A1,	B1,	C1,	D1,	E1,
A12,A13,	B12,B13,	C12,C13,	D12,D13,	E12,E13,
A14,A15,	B14,B15,	C14,C15,	D14,D15,	E14,E15,
A123,A124,	B123,B124,	C123,C124,	D123,D124,	E123,E124,
A125,A134,	B125,B134,	C125,C134,	D125,D134,	E125,E134,
A145,A135,	B145,B135,	C145,C135,	D145,D135,	E145,E135,
A1234,A1345,	B1234,B1345,	C1234,C1345,	D1234,D134,	E1234,E1345,
A1245,A1235,	B1245,B1235,	C1245,C1235,	D1245,D123,	E1245,E1235,
A12345	B12345	C12345	D12345	E12345

Table 3.2: Placement phase(cache content of user1)

2. **Cache content of user2**

Fragments of file A	Fragments of file B	Fragments of file C	Fragments of file D	Fragments of file E
A1,	B1,	C1,	D1,	E1,
A12,A23,	B12,B23,	C12,C23,	D12,D23,	E12,E23,
A24,A25,	B24,B25,	C24,C25,	D24,D25,	E24,E25,
A123,A124,	B123,B124,	C123,C124,	D123,D124,	E123,E124,
A125,A234,	B125,B234,	C125,C234,	D125,D234,	E125,E234,
A235,A245,	B235,B245,	C235,C245,	D235,D245,	E235,E245,
A1234,A2345,	B1234,B2345,	C1234,C2345,	D1234,D2345,	E1234,E2345,
A1245,A1235,	B1245,B1235,	C1245,C1235,	D1245,D1235,	E1245,E1235,
A12345	B12345	C12345	D12345	E12345

Table 3.3: Placement phase(cache content of user2)

3. **Cache content of user3**

Fragments of file A	Fragments of file B	Fragments of file C	Fragments of file D	Fragments of file E
A3,	B3,	C3,	D3,	E3,
A13,A23,	B13,B23,	C13,C23,	D13,D23,	E13,E23,
A34,A35,	B34,B35,	C34,C35,	D34,D35,	E34,E35,
A123,A234,	B123,B234,	C123,C234,	D123,D234,	E123,E234,
A235,A345,	B235,B345,	C235,C345,	D235,D345,	E235,E345,
A134,A135,	B134,B135,	C134,C135,	D134,D135,	E134,E135,
A1234,A2345,	B1234,B2345,	C1234,C2345,	D1234,D2345,	E1234,E2345,
A1345,A1235,	B1345,B1235,	C1345,C1235,	D1345,D1235,	E1345,E1235,
A12345	B12345	C12345	D12345	E12345

Table 3.4: Placemnet phase(cache content of user3)

4. **Cache content of user4**

Fragments of file A	Fragments of file B	Fragments of file C	Fragments of file D	Fragments of file E
A4, A14,A24, A34,A45, A124,A234, A345,A134, A145,A245, A1234,A2345, A1345,A1245, A12345	B4, B14,B24, B34,B45, B124,B234, B345,B134, B145,B245, B1234,B2345, B1345,B1245, B12345	C4, C14,C24, C34,C45, C124,C234, C345,C134, C145,C245, C1234,C2345, C1345,C1245, C12345	D4, D14,D24, D34,D45, D124,D234, D345,D134, D145,D245, D1234,D2345, D1345,D1245, D12345	E4, E14,E24, E34,E45, E124,E234, E345,E134, E145,E245, E1234,E2345, E1345,E1245, E12345

Table 3.5: Placemnet phase(cache content of user4)

5. **Cache content of user5**

Fragments of file A	Fragments of file B	Fragments of file C	Fragments of file D	Fragments of file E
A5, A15,A25, A35,A45, A125,A235, A345,A145, A245,A135, A2345,A1345, A1245,A1235, A12345	B5, B15,B25, B35,B45, B125,B235, B345,B145, B245,B135, B2345,B1345, B1245,B1235, B12345	C5, C15,C25, C35,C45, C125,C235, C345,C145, C245,C135, C2345,C1345, C1245,C1235, C12345	D5, D15,D25, D35,D45, D125,D235, D345,D145, D245,D135, D2345,D1345, D1245,D1235, D12345	E5, E15,E25, E35,E45, E125,E235, E345,E145, E245,E135, E2345,E1345, E1245,E1235, E12345

Table 3.6: Placemnet phase(cache content of user5)

- **Content Delivery** Delivery phase starts with s=K, here, K=5 and we have considered worst case scenario.

  1. First iteration s=5, server will transmit coded signal useful simultaneously for 5 users. Server will multicast

     A2345 $\oplus$ B1345 $\oplus$ C1245 $\oplus$ D1235$\oplus$ E1234

  2. Second iteration s=4, server will transmit coded signal useful simultaneously for 4 users. Server will multicast-

     A234 $\oplus$ B134 $\oplus$ C124 $\oplus$ D123

     A235 $\oplus$ B135 $\oplus$ C125 $\oplus$ E123

     A345 $\oplus$ C145 $\oplus$ D135 $\oplus$ E134

     A245 $\oplus$ B145 $\oplus$ D125 $\oplus$ E124

     B345 $\oplus$ C245 $\oplus$ D235 $\oplus$ E234

3. Third iteration s=3, server will transmit coded signal useful simultaneously for 3 users. Server will multicast-

   A23 $\oplus$ B13 $\oplus$ C12

   A24 $\oplus$ B14 $\oplus$ D12

   A25 $\oplus$ B15 $\oplus$ E12

   A34 $\oplus$ C14 $\oplus$ D13

   A35 $\oplus$ C15 $\oplus$ E13

   A45 $\oplus$ D15 $\oplus$ E14

   B34 $\oplus$ C24 $\oplus$ D23

   B35 $\oplus$ C25 $\oplus$ E23

   B45 $\oplus$ D25 $\oplus$ E24

   C45 $\oplus$ D35 $\oplus$ E34

4. Fourth iteration s=2, user cooperation will eliminate need of transmission from server.

5. Fifth iteration s=1, server will transmit coded signal useful only for 1 users. Server will unicast A0, B0, C0, D0, E0 to user1, user2, user3, user4 and user5.

Now, calculating the rate,

- When s=5, as BS transmits 1 coded message of 5 file fragments which are 1/32th of corresponding files. So the rate is = 1/32. [as 1 coded transmission]

- When s=4, as BS transmits 5 coded message of 4 file fragments which are 1/32th of corresponding files. So the rate is = 5/32. [as 5 coded transmission]

- When s=3, as BS transmits 10 coded message of 3 file fragments which are 1/32th of corresponding files. So the rate is = 10/32. [as 10 coded transmission]

∗ When s=1, as BS transmits 5 file fragments which are 1/32th of corresponding files. So the rate is = 5/32. [as 5 coded transmission]

So, the rate will be

R_{UC} (M)= 1/32 + 5/32 +10/32 + 5/32 = 21/32

Now,calculating rate by using the equation generated for user cooperation R_{UC} (M)=5×(1-M/5)^5 + $^5C_3$ ×(M/5)^2 ×(1-M/5)^3 + $^5C_4$ ×(M/5)^3 × (1-M/5)^2 + $^5C_5$ × (M/5)^4 × (1-M/5)

= 5× (1-2.5/5)^5 + 10×(2.5/5)^2 ×(1-2.5/5)^3 + 5× (2.5/5)^3 ×(1-2.5/5)^2 + 1× (2.5/5)^4 × (1-2.5/5)

=5×(1-1/2)^5 + 10×(1/2)^2 × (1-1/2)^3 + 5 × (1/2)^3 × (1-1/2)^2 + 1 × (1/2)^4 × (1-1/2)

=5×(1/2)^5 + 10×(1/2)^2 ×(1/2)^3 + 5× (1/2)^3 × (1/2)^2 + 1 × (1/2)^4 × (1/2)

=5×(1/32) + 10 × (1/4) × (1/8) + 5× (1/8) × (1/4) + 1 × (1/16) × (1/2)

=5/32 + 10/32 + 5/32 + 1/32

= 21/32

From above calculation, we can observe that, rate calculated by using the new generalized equation is equal to the rate calculated manually. It provides the proper justification of our equation.

3.6 Chapter Summery

In this section we have given an overview of our proposed caching scheme. The proposed algorithm was shown in flowcharts. We have tried to present our work with aid of graphical representation. Later we have developed a generalized equation step by step to quantify the delivery rate of proposed algorithm. The section has been concluded with a elaborated verification regarding our proposed algorithm.

Chapter 4

Results and Discussions

4.1 Introduction

In this section we have evaluated our proposed algorithm by comparing it with other existing algorithm. Firstly, we have shown the comparison between existing algorithms. Then we have compared our proposed algorithm with the existing decentralized coding approach. We have compared our proposed algorithm with the uncoded caching approach used in existing IoT network. Finally, we have compared our proposed algorithm with all the existing algorithm.

We have focused on the following parameters to derive our comparisons:

1. Number of user nodes, K

2. Number files in BS, N

3. Cache memory size, M

4.Bottleneck rate, R

4.2 Comparison Between Existing Algorithms

We have mentioned some existing caching approaches in the literature review chapter. We have also mentioned that we have chosen a suitable approach to invoke it into the IoT network. We have generated some graphs to compare their performance and relied on those graphs to take decision about choosing the right caching approach. Here, these curves are generated by plotting equations 2.1, 2.2, 2.3 where M is a function of R and N, K are kept constant.

1. **When N=2 ,K=2**

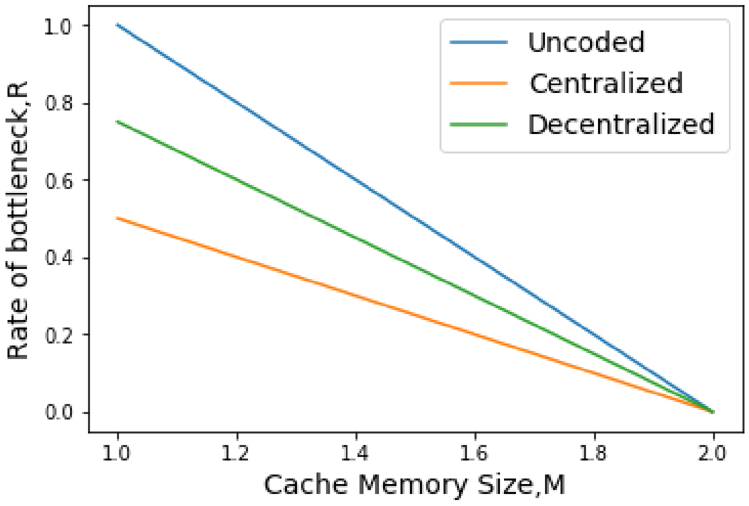


Figure 4.1: Comparison Between Existing Algorithms (N=K=2)

Here, we have taken the value N=2 and K=2 that means server has 2 files and there are 2 user nodes in the corresponding IoT network. The horizontal axis shows the cache memory size, M and the vertical axis shows the bottleneck link rate, R. The blue curve represents the rate achieved by the uncoded caching approach. The orange curve represents the rate achieved by the centralized coded caching and the green curve represents the rate achieved by the decentralized coded caching approach. We can see that, for a network with 2 users the curves are linear. We can say that change of rate with respect to cache memory size is linear. Rate decreases with the increase of cache memory size. If we take a random point on x axis, we can observe the rates of different approaches. We know that smaller slope indicates reduced rate. We can clearly see that centralized coded caching offers the best rate as it has smaller slope and it offers smaller R for any point on x axis.

2. **When N=2,K=2**

Here, we have taken the value N=2 and K=2 that means server has 2 files and there are 2 user nodes in the corresponding IoT network. The horizontal axis shows the normalized cache size, (M/N) and the vertical axis shows the bottleneck link rate, R. The blue curve represents the rate achieved by the uncoded caching approach. The orange curve represents the rate

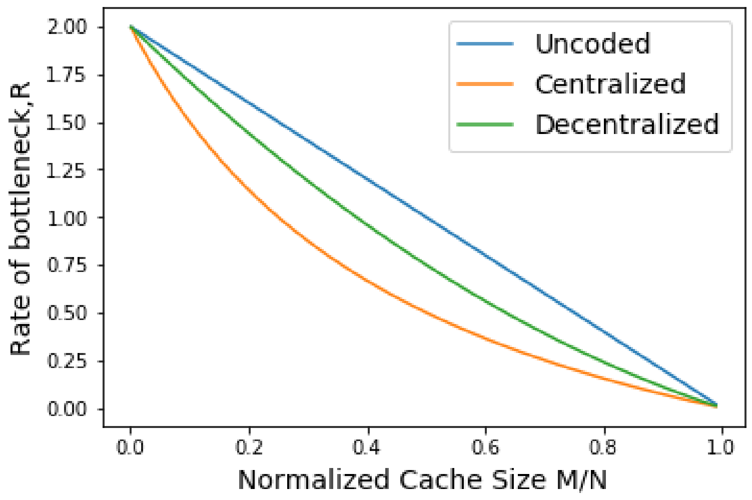


Figure 4.2: Comparison Between Existing Algorithms (Normalized cache Size)

achieved by the centralized coded caching and the green curve represents the rate achieved by the decentralized coded caching approach. Here the blue curve represents linear decrease of the rate. But both orange and green curve shows non-linear decrease of rate. We can say both centralized and decentralized coded caching gives better rate then uncoded as both have better slope then the blue curve. The centralized caching has smaller slope as the slope is negative and reciprocal of second degree. So, it has the best rate among all of the existing approaches.

3. **When N=3 ,K=3**

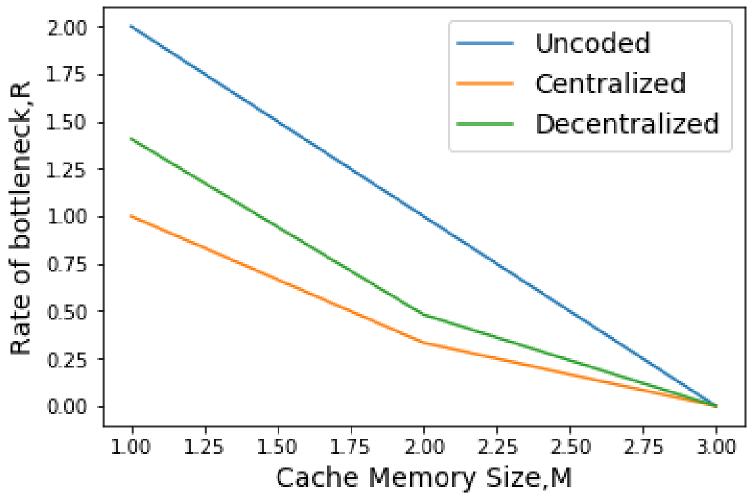


Figure 4.3: Comparison Between Existing Algorithms (N=K=3)

Here, we have taken the value N=3 and K=3. The horizontal axis shows the cache memory size, M and the vertical axis shows the bottleneck link rate, R. The blue curve represents the rate achieved by the uncoded caching approach. The orange curve represents the rate achieved by the centralized coded caching and the green curve represents the rate achieved by the decentralized coded caching approach. The rate achieved by the uncoded approach is as usual linear even for an IoT network with 3 users. The green curve changes slope at a certain point 2.00 of x axis. It happens because rate of decentralized coded caching decreases more rapidly with the increment of the size of cache memory before this certain value. So it has different slope values. Same goes for green curve also. But as centralized coded caching has smaller slope so, it shows the best rate amongst the all approaches.

4. **When N=100,K=20**

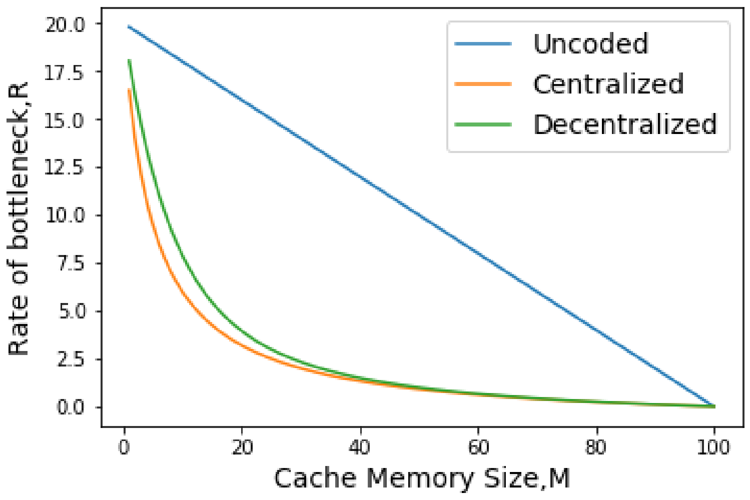


Figure 4.4: Comparison Between Existing Algorithms (N=100 and K=20)

Here, we have taken the value N=100 and K=20. The horizontal axis shows the cache memory size, M and the vertical axis shows the bottleneck link rate, R. The blue curve represents the rate achieved by the uncoded caching approach. The orange curve represents the rate achieved by the centralized coded caching and the green curve represents the rate achieved by the decentralized coded caching approach. We can see that for increased number of users and files, centralized and decentralized coded caching shows rates that are very closer to each other. Here decrease rate of uncoded caching is constant as the slope of this curve doesn't depend on M. But both orange and green curves show rapid decrease with the increase of the cache memory size.

From above graphs, it is quite clear that decentralized coded caching has better rate then an uncoded caching approach. The rate of this approach is closer to centralized coded caching. As we have mentioned before, decentralized coded caching doesn't require central coordination system and gives us some fleixibilities also. That's why we have chosen decentralized coded caching approach.

4.3 Comparison Between Existing Decentralized Coded Caching and Proposed Caching Scheme

Our proposed algorithm is a modified version of existing decentralized coded caching approach. As we have introduced user cooperation in this algorithm to make it suitable for an IoT network. It should give us better rate then the existing one. Now we will compare it's rate with the existing one. Here these curves are generated by plotting equations 2.3 and 3.1 where M is a function of R and N, K are kept constant.

1. **When N=2 and K=2**

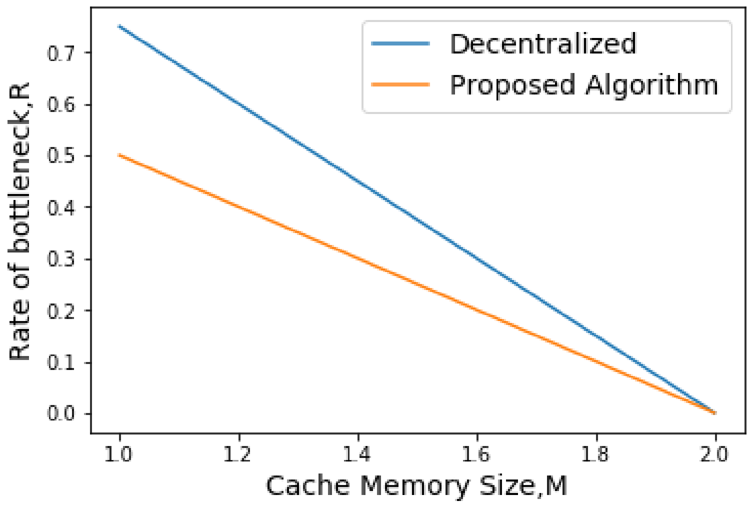


Figure 4.5: Comparison Between Decentralized Caching and Proposed Algorithm (N=K=2)

Here, we have taken the value N=2 and K=2 that means server has 2 files and there are 2 user nodes in the corresponding IoT network. The horizontal axis shows the cache memory size, M and the vertical axis shows the bottleneck link rate, R. The blue curve represents the rate achieved by decentralized coded caching approach. The orange curve represents the rate achieved by the proposed coded caching approach. The both curves are linear. We can say that change of rate with respect to cache memory size is linear. Rate decreases with the increase of cache memory size. If we take a random point on x axis, we can observe the rates of the both approaches. We can clearly see that our proposed algorithm has smaller slope. We know that smaller slope means reduced rate. So, it offers better R for any point on x axis.

2. **When N=3 and K=3**

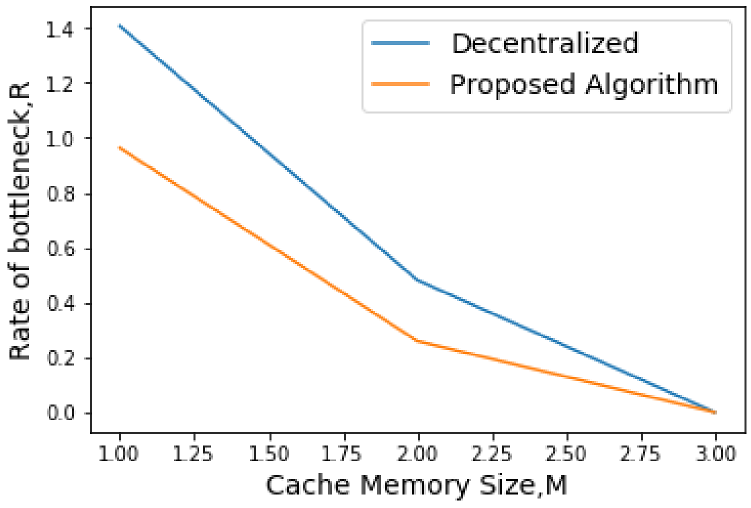


Figure 4.6: Comparison Between Decentralized Caching and Proposed Algorithm (N=K=3)

Here, we have taken the value N=3 and K=3 that means server has 3 files and there are 3 user nodes in the corresponding IoT network. The horizontal axis shows the cache memory size,M and the vertical axis shows the bottleneck link rate, R. The blue curve represents the rate achieved by decentralized coded caching approach. The orange curve represents the rate achieved by the proposed coded caching approach. We can see that both curves are steeper before the value 2.00 along x axis. It occurs because the decrease rate of R is more rapid before this point. The decrement of R with the respect to M is still linear after this certain point but the slope value is different and the curves are less steep. We know that small value of slope indicates reduced rate. As our proposed caching scheme has smaller values of slopes than the decentralized caching approach so it offers better rate.

3. **When N=4 and K=4**

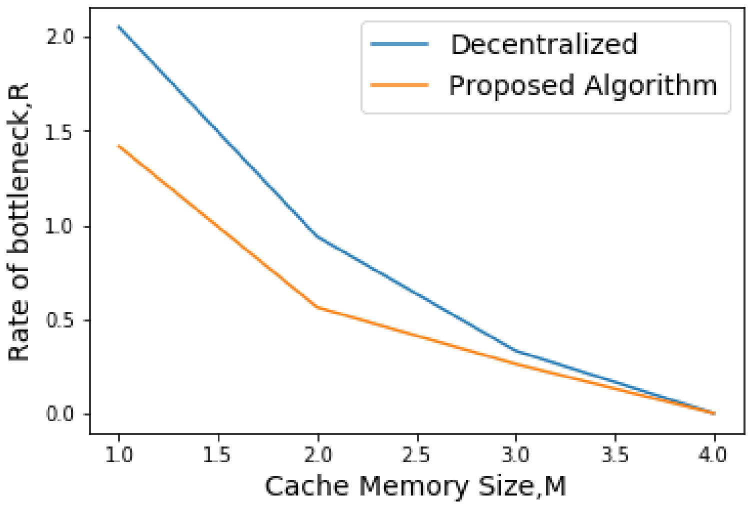


Figure 4.7: Comparison Between Decentralized Caching and Proposed Algorithm (N=K=4)

Here, we have taken the value N=4 and K=4. The horizontal axis shows the cache memory size, M and the vertical axis shows the bottleneck link rate, R. The blue curve represents the rate achieved by decentralized coded caching approach. The orange curve represents the rate achieved by the proposed coded caching approach. The blue curve can be divided into three distinct regions. All of this three regions represents linear decrement but the rate of decrement is different for each region as slopes are different. The orange curve can be divided into two distict regions. At the first region, the both curves show most steep form. That means in this region rate decreases more rapidly. After the value 2.0 along x axis, both curves become less steep. As the curve of our proposed approach gives us smaller value of slope at any point than the decentralized approach so we can say it gives us better rate.

4. **When N=20 and K=20**

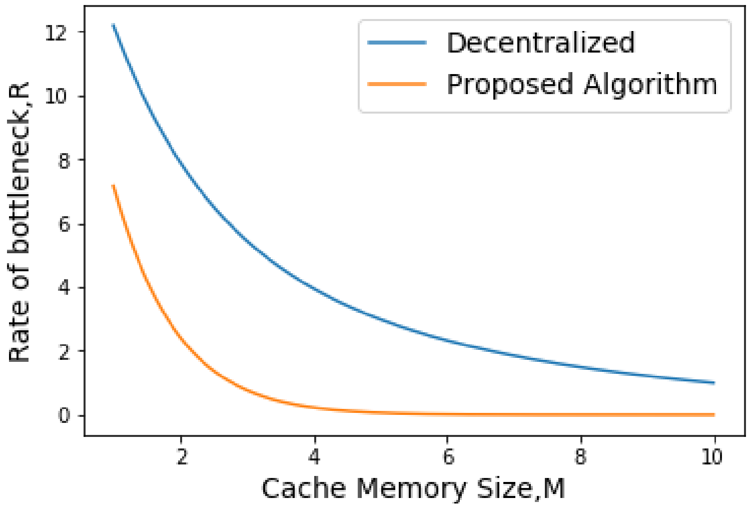


Figure 4.8: Comparison Between Decentralized Caching and Proposed Algorithm (N=K=20)

Here, we have taken the value N=20 and K=20. The horizontal axis shows the cache memory size, M and the vertical axis shows the bottleneck link rate, R. The blue curve represents the rate achieved by decentralized coded caching approach. The orange curve represents the rate achieved by the proposed coded caching approach. Here, the both curves are higher degree polynomial. But they have different slope values. As our proposed caching approach has the smallest slope that means it has a rate less than the decentralized caching approach. So, it provides the better rate.

In above graphs performance of both caching scheme has been observed. We can see that our proposed algorithm gives a better rate than the existing decentralized approach. Here we have considered multiple scenarios with varying number of users and varying number of files. In each case our algorithm shows an improved delivery rate.

4.4 Comparison Between Caching approach of Existing IoT Network and Proposed Scheme

We have proposed a caching approach in which user cooperation has been introduced in decentralized coded caching. It can replace the uncoded caching used in the IoT network described in [1]. Now we will compare our proposed caching scheme with the uncoded caching approach of existing IoT network. Here these curves are generated by plotting equations 2.1 and 3.1 where R is a function of M and N, K are kept constant.

1. **When N=2 and K=2**

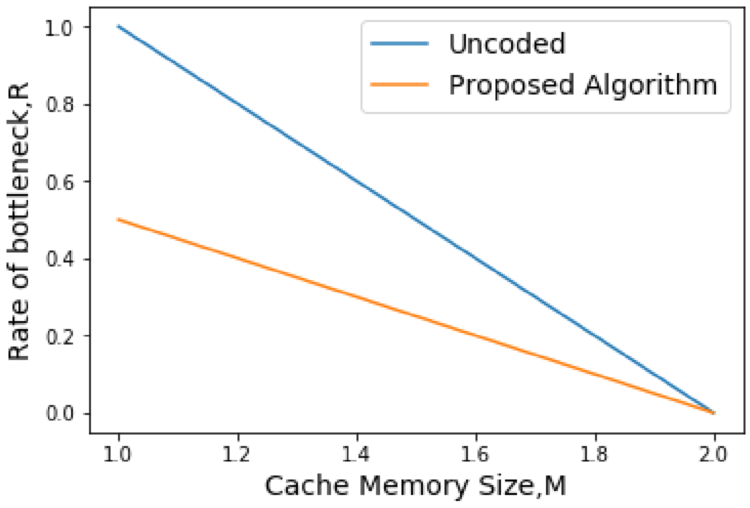


Figure 4.9: Comparison Between Uncoded Caching and Proposed Algorithm (N=K=2)

Here, we have taken the value N=2 and K=2 that means server has 2 files and there are 2 user nodes in the corresponding IoT network. The horizontal axis shows the cache memory size, M and the vertical axis shows the bottleneck link rate, R. The blue curve represents the rate achieved by uncoded caching approach used in the IoT network. The orange curve represents the rate achieved by the proposed coded caching approach. The both curves are linear. We can say that change of rate with respect to cache memory size is linear. Rate decreases with the increase of cache memory size. If we take a random point on x axis, we can observe the rates of the both approaches. We can clearly see that our proposed algorithm has smaller slope. We know that smaller slope means reduced rate. So, our proposed caching approach has a better rate than the uncoded one.

2. **When N=3 and K=3**

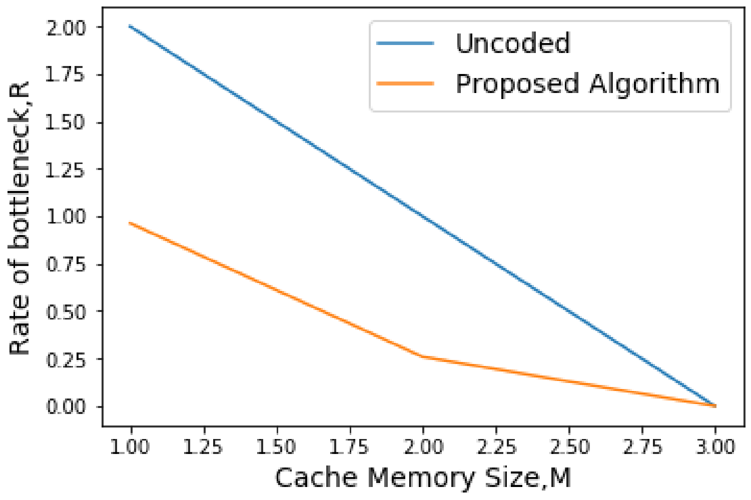


Figure 4.10: Comparison Between Uncoded Caching and Proposed Algorithm (N=K=3)

Here, we have taken the value N=3 and K=3. The horizontal axis shows the cache memory size, M and the vertical axis shows the bottleneck link rate, R. The blue curve represents the rate achieved by uncoded caching approach used in the IoT network. The orange curve represents the rate achieved by the proposed coded caching approach. The blue curve is as usual linear. The orange curve changes slope at a certain point 2.00 of x axis. It happens because rate of proposed coded caching decreases more rapidly with the increment of the size of cache memory before this certain value. So it has different slope values. Our proposed coded caching has smaller slope so it shows the better rate.

3. **When N=4 and K=4**

Here, we have taken the value N=4 and K=4 to observe the rates of the both approaches. The horizontal axis shows the cache memory size, M and the vertical axis shows the bottleneck link rate, R. The blue curve represents the rate achieved by uncoded caching approach used in the IoT network. The orange curve represents the rate achieved by the proposed coded caching approach. The blue curve is as usual linear but we can divide the orange curve into two distinct regions. This two regions represents linear decrease but the rate of decrease is different for each region as slopes are different. At the first region, rate decreases more rapidly. After the value 2.0 along x

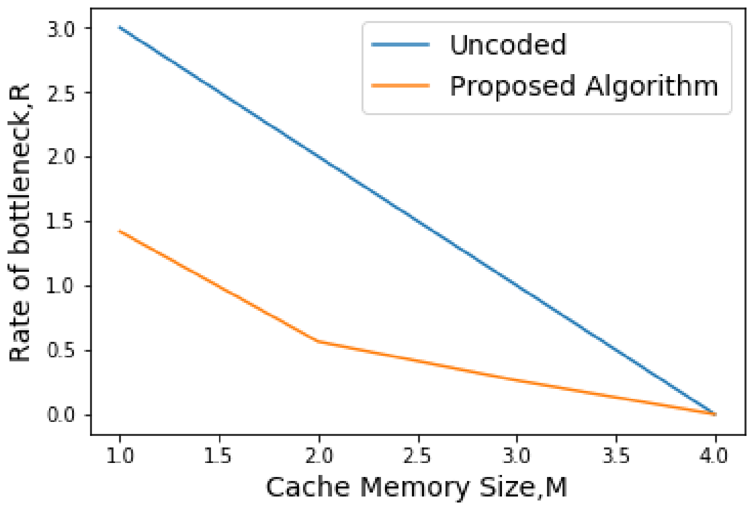


Figure 4.11: Comparison Between Uncoded Caching and Proposed Algorithm (N=K=4)

axis, this curve become less steep. As the curve of our proposed approach gives us smaller value of slope at any point than the decentralized approach so we can say it gives us better rate.

4. **When N=20 and K=20**

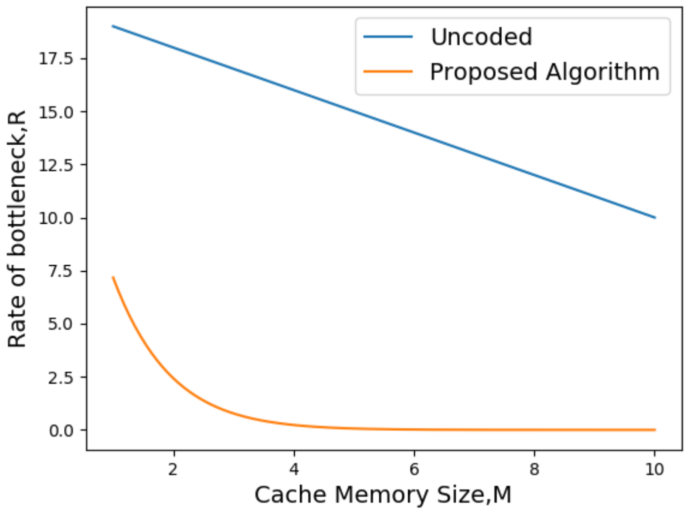


Figure 4.12: Comparison Between Uncoded Caching and Proposed Algorithm (N=K=20)

Here, we have taken the value N=20 and K=20 to observe the rates of the both approaches. The horizontal axis shows the cache memory size, M and the vertical axis shows the bottleneck link rate, R. The blue curve represents the rate achieved by uncoded caching approach used in the IoT network. The orange curve represents the rate achieved by the proposed coded caching approach. For a large number of users in the network, we can see a huge difference between the rates of both approaches. The blue curve is linear. On the other hand, the orange curve is non-linear but value of its slope is far less then the uncoded caching approach. For a IoT network with 20 user nodes, the performance of our proposed caching approach is undoubtedly superior.

In above graphs performance of both uncoded and our proposed caching scheme have been observed. We can see that rate of our proposed scheme is better then the uncoded scheme. If our proposed scheme is deployed in The IoT network instead of uncoded caching, it will give a better rate than the existing approach. Additionally it provides the functionality of decentralized coded caching. So we can get a caching system with greater flexibility and better delivery rate. Here we have considered multiple scenarios with varying number of users and files. In each case our algorithm shows an improved delivery rate.

4.5 Comparison Between Existing and Proposed Caching Schemes

In this section we have compared our proposed caching approach with all other approaches. Our proposed caching scheme should have better rate then uncoded and decentralized caching approach. We already know that centralized caching has a better rate but less flexibility. Our target was to make the rate almost same to centralized caching but to provide flexibility of decentralized caching. And our algorithm has accommodated user cooperation to make it suitable for the IoT network. We can observe performance of our proposed scheme from the generated graphs listed below.

1. **When N=2 and K=2**

   Here, we have observed the rate of all approaches including the proposed one for varying cache size. When M=1, we can see that our proposed approach works better then both decentralized coded caching and the uncoded caching used in the IoT network. But when M=1.5, it shows a significantly

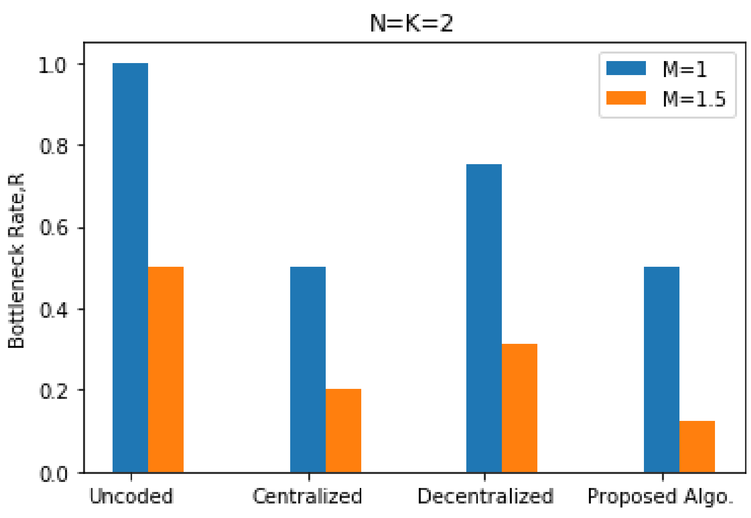


Figure 4.13: Comparison Between Existing Caching Schemes and Proposed Scheme (N=K=2)

good performance as it achieves a rate less then centralized coded caching also. The overall rate of our proposed algorithm is quite satisfactory.

2. **When N=3 and K=3**

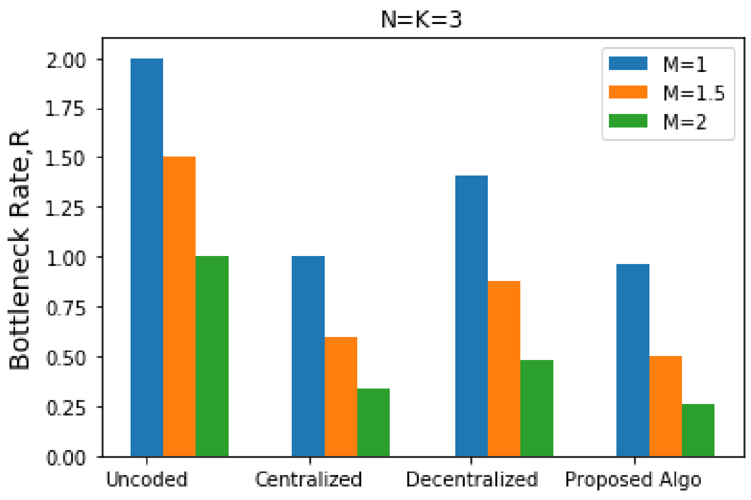


Figure 4.14: Comparison Between Existing Caching Schemes and Proposed Scheme (N=K=3)

Here, we have observed the rate of all approaches including the proposed one for varying cache size, M. When M=1, we can see that our proposed approach works better then both decentralized coded caching and the uncoded caching used in the IoT network. When M=1.5, it achieves slightly better rate than centralized coded caching. But when M=2, it shows a good performance as it offers a rate less then centralized coded caching.

The overall rate of our proposed algorithm is better than the existing caching approaches.

3. **When N=4 and K=4**

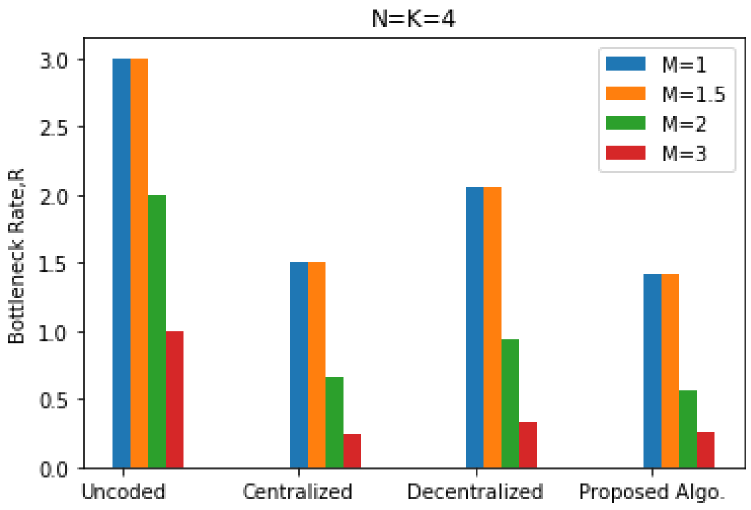


Figure 4.15: Comparison Between Existing Caching Schemes and Proposed Scheme (N=K=4)

Here, we have observed the rate of all approaches including the proposed one for varying cache size, M. Performance of all the approaches are same for values M=1 and M=1.5 We can see that our proposed approach works better then both decentralized coded caching and the uncoded caching used in the IoT network. When M=2, it even achieves better rate than centralized coded caching. But when M=3, it shows a good performance as it offers a rate almost same as centralized coded caching. The overall rate of our proposed algorithm is better than the existing caching approaches.

We can see that, Our proposed Algorithm works much better then uncoded caching for each value of M. The rate is almost same to centralized caching approach. We can also observe that when cache size(M) is half of the file size(N), it gives us better rate then centralized caching. So, according to these graphs we can say our initial assumptions about the rate of the proposed schemes are realistic.

4.6 Chapter Summery

In this section we have presented the performance evaluation of our proposed scheme. We have observed the delivery rate of our proposed work compared to existing approaches. We have seen that our proposed coded caching approach is suitable for an IoT network and it provides a better delivery rate.

Chapter 5

Conclusion

5.1 Conclusion

In this thesis we have presented a decentralized coded caching approach which allows user cooperation. We have modified the existing one to make it appropriate for an IoT network. By replacing the existing uncoded caching approach in IoT by our proposed caching scheme we have created multicasting opportunity which was not exploited before in IoT network. We have also improved the delivery rate. By invoking decentralized coded caching in IoT we have eliminated the need of any central coordination system to manage the placement phase. Thus a flexible and better caching approach has been introduced in IoT network.

5.2 Future Work

Some future improvements will make the thesis more perfect. These are listed below

- User cooperation should be quantified.
- Complexities due to increase in number of user nodes should be identified.
- Adverse effecs of user cooperation should be investigated.

References

[1] Y. Ai, L. Wang, Z. Han, P. Zhang and L. Hanzo, 'Social networking and caching aided collaborative computing for the internet of things,' *IEEE Communications Magazine*, vol. 56, no. 12, pp. 149–155, 2018 (cit. on pp. 1, 5, 15, 45).

[2] A. Zanella, N. Bui, A. Castellani, L. Vangelista and M. Zorzi, 'Internet of things for smart cities,' *IEEE Internet of Things journal*, vol. 1, no. 1, pp. 22–32, 2014 (cit. on pp. 1, 5).

[3] M. A. Maddah-Ali and U. Niesen, 'Fundamental limits of caching,' *IEEE Transactions on information theory*, vol. 60, no. 5, pp. 2856–2867, 2014 (cit. on pp. 1, 2, 5, 6, 9).

[4] V. K. Kumar, B. K. Rai and T. Jacob, 'Towards the exact rate memory tradeoff in coded caching,' in *2019 National Conference on Communications (NCC)*, IEEE, 2019, pp. 1–6 (cit. on p. 1).

[5] A. Whitmore, A. Agarwal and L. Da Xu, 'The internet of things—a survey of topics and trends,' *Information systems frontiers*, vol. 17, no. 2, pp. 261–274, 2015 (cit. on p. 5).

[6] L. Atzori, A. Iera and G. Morabito, 'The internet of things: A survey,' *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010 (cit. on p. 5).

[7] U. Niesen and M. A. Maddah-Ali, 'Coded caching for delay-sensitive content,' in *2015 IEEE International Conference on Communications (ICC)*, IEEE, 2015, pp. 5559–5564 (cit. on p. 6).

[8] M. A. Maddah-Ali and U. Niesen, 'Decentralized coded caching attains order-optimal memory-rate tradeoff,' *IEEE/ACM Transactions On Networking*, vol. 23, no. 4, pp. 1029–1040, 2014 (cit. on pp. 6, 7, 9, 11, 12, 16, 28).

[9] M. M. Amiri, Q. Yang and D. Gündüz, 'Coded caching for a large number of users,' in *2016 IEEE Information Theory Workshop (ITW)*, IEEE, 2016, pp. 171–175 (cit. on pp. 8, 9).

[10] A. Sengupta, R. Tandon and T. C. Clanc, 'Layered caching for heterogeneous storage,' in *2016 50th Asilomar Conference on Signals, Systems and Computers*, IEEE, 2016, pp. 719–723 (cit. on p. 8).

[11] A. M. Ibrahim, A. A. Zewail and A. Yener, 'Centralized coded caching with heterogeneous cache sizes,' in *2017 IEEE Wireless Communications and Networking Conference (WCNC)*, IEEE, 2017, pp. 1–6 (cit. on p. 8).

[12] ——, 'Coded caching for heterogeneous systems: An optimization perspective,' *IEEE Transactions on Communications*, vol. 67, no. 8, pp. 5321–5335, 2019 (cit. on p. 8).

[13] K. Zhang and C. Tian, 'From uncoded prefetching to coded prefetching in coded caching systems,' in *2018 IEEE International Symposium on Information Theory (ISIT)*, IEEE, 2018, pp. 2087–2091 (cit. on p. 8).

[14] J. Gómez-Vilardebó, 'Fundamental limits of caching: Improved bounds with coded prefetching,' *arXiv preprint arXiv:1612.09071*, 2016 (cit. on p. 8).

[15] A. M. Ibrahim, A. A. Zewail and A. Yener, 'Benefits of coded placement for networks with heterogeneous cache sizes,' in *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, IEEE, 2018, pp. 1604–1608 (cit. on pp. 8, 9).

[16] D. Cao, D. Zhang, P. Chen, N. Liu, W. Kang and D. Gunduz, 'Coded caching with heterogeneous cache sizes and link qualities: The two-user case,' in *2018 IEEE International Symposium on Information Theory (ISIT)*, IEEE, 2018, pp. 1545–1549 (cit. on p. 9).

[17] A. M. Ibrahim, A. A. Zewail and A. Yener, 'Coded placement for systems with shared caches,' (cit. on p. 9).

[18] M. A. Maddah-Ali and U. Niesen, 'Coding for caching: Fundamental limits and practical challenges,' *IEEE Communications Magazine*, vol. 54, no. 8, pp. 23–29, 2016 (cit. on pp. 10, 12, 14, 16).

[19] U. Niesen and M. A. Maddah-Ali, 'Coded caching with nonuniform demands,' *IEEE Transactions on Information Theory*, vol. 63, no. 2, pp. 1146–1158, 2016 (cit. on p. 14).

[20] R. Pedarsani, M. A. Maddah-Ali and U. Niesen, 'Online coded caching,' *IEEE/ACM Transactions on Networking*, vol. 24, no. 2, pp. 836–845, 2015 (cit. on p. 14).

[21] Y. Wang, J. Wu and M. Xiao, 'Hierarchical cooperative caching in mobile opportunistic social networks,' in *2014 IEEE Global Communications Conference*, IEEE, 2014, pp. 411–416 (cit. on p. 15).

[22] D. Wu, B. Liu, Q. Yang and R. Wang, 'Social-aware cooperative caching mechanism in mobile social networks,' *Journal of Network and Computer Applications*, vol. 149, p. 102 457, 2020 (cit. on p. 15).

[23] L. Wang, H. Wu, Z. Han, P. Zhang and H. V. Poor, 'Multi-hop cooperative caching in social iot using matching theory,' *IEEE Transactions on Wireless Communications*, vol. 17, no. 4, pp. 2127–2145, 2017 (cit. on p. 15).