UNIVERSITY OF CALIFORNIA
SANTA CRUZ

CODED CACHING IN WIRELESS NETWORKS AND STORAGE SYSTEMS

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

ELECTRICAL ENGINEERING

by

Mohsen Karimzadeh Kiskani

September 2017

The Dissertation of Mohsen Karimzadeh Kiskani
is approved:

Professor Hamid R. Sadjadpour, Chair

Professor Donald Wiberg

Reza Rahimi, Ph.D.

Tyrus Miller
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

List of Tables

Abstract

Coded Caching in Wireless Networks and Storage Systems

by

Mohsen Karimzadeh Kiskani

Coded caching in wireless networks and storage systems is studied. Caching based on two types of codes, index codes and fountain codes, is investigated in terms of network throughput and security. It is shown that index coding can significantly increase the multicast transmission rate in Information Centric Networks (ICN). Also, it is proved that index codes can be efficiently used to increase the multicasting transmission rate in cellular networks. It is proved that a simple graph coloring-based algorithm for index coding achieves order optimal capacity gains both for cellular and ICN networks. A new decentralized caching scheme based on Random Linear Fountain (RLF) codes is then introduced and it is shown that RLF-based coded caching performs close to optimal in terms of reducing the average number of transmission hops in wireless ad hoc and cellular networks. Therefore, considerable capacity gains can be achieved using RLF-based coded caching in wireless networks. It is shown that the RLF codes can significantly reduce the overcaching in wireless networks. Further, it is shown that using coded caching based on RLF codes we can achieve asymptotic perfect secrecy. In the limiting case of large number of coded files, the conditions of Shannon secrecy theorem are met and the problem can be modelled by a Shannon cipher system which is perfectly secure. Finally, a new storage policy based on RLF codes for storage systems

along with a Private Information Retrieval (PIR) scheme for these systems is proposed and it is proved that perfect privacy and secrecy is achievable in these systems.

To my parents and my wife,

for their love, and support.

Acknowledgments

There are many individuals who helped me to make this dissertation possible. This work is not just the result of my efforts but also reflects the mentoring and support I have received.

First and foremost, I want to thank my adviser Prof. Hamid Sadjadpour, for his great ideas, wise comments, and generous support and guidance through my long graduate school journey. His enthusiasm and true knowledge influenced me to be a better researcher.

I also thank my committee members, Prof. Donald Wiberg and Dr. Reza Rahimi, for their encouragement and invaluable comments and suggestions. Their thoughtful insights as well as their sincerity motivated me and taught me how to stay positive.

I lastly thank our Lab members for helpful discussions and support. I especially want to acknowledge Bita Azimdoost for her helpful collaboration, and Jose Armando Oviedo for his support and encouragement.

Chapter 1

Introduction

Caching has been a major area of research in recent years after the seminal work of Maddah-Ali et. al. in [72]. Many researchers have tried to extend the results in [72] for different scenarios in broadcast channels [38, 49, 73, 80]. In all of these papers during the cache placement phase, only uncoded contents or uncoded parts of contents are stored in the caches. Later during the content delivery phase, the base station broadcasts coded contents (linear combination of multiple contents) to users such that they can decode their files simultaneously. In our work, we propose coding techniques to boost the performance of caching. Therefore, our proposed coded caching techniques are fundamentally different from the notion of coded caching in references like [38, 49, 73, 80]. In our work, we used index coding and fountain coding as the main coding techniques to apply in caching systems.

In chapter 2 we propose a new architecture for Information Centric Networks (ICN) which takes advantage of index coding to increase the multicast transmission

rates. Index coding [9, 10] focuses on broadcast channels in which the transmitter utilizes coding in order to reduce the number of transmissions. ICN focuses on the content delivery without any considerations on where the content is obtained. We show that a significant number of transmissions can be saved through the use of index codes in ICNs. This is mainly due to the Zipfian-like web content request popularity distribution in ICN [15]. This content popularity distribution implies that few popular contents are widely requested by the network users. We assume that users store their requested contents and therefore, routers can multicast multiple requests by taking advantage of coding to reduce the total number of transmissions.

In chapter 3 we will show the benefits of index coding in wireless cellular networks which are based on the idea of femtocaching proposed by Golrezaei et. al. in [37]. The proposed technique in [37] requires deployment of large number of femtocaches in order to cache the contents locally and retrieve them when needed. We propose to use a multihop transmission scheme which significantly reduces the femtocache deployment costs compared to [37]. Further, we show that significant capacity gains can be achieved through the use of index codes. The optimal index coding solution is an NP-Hard problem [64]. We propose a simple heuristic to perform the task of index coding and we will show that this heuristic which is based on graph coloring is asymptotically capable of achieving maximum index coding capacity.

In chapter 4, we proposed a novel decentralized coded caching scheme based on Random Linear Fountain (RLF) codes. In this technique, each user independently caches a random combination of all the other files in a decentralized manner. Re-

dundant caching is avoided by storing a random bitwise XOR combination of popular contents. This approach increases the network throughput capacity and does not suffer from over-caching problem of uncoded caching. We propose that during the cache placement phase, the contents are randomly combined and cached in network nodes. We show that this coded caching approach performs near optimal in terms of the average number of hops to retrieve a content and hence, it can significantly increase the network throughput capacity. This makes the proposed coded cache placement very suitable in practical systems where UTs have small storage capability compared to the total number of contents in the network. We prove that the proposed decentralized coded content caching increases the capacity of cellular networks by a factor of $(\log(n))^2$ compared to decentralized uncoded caching.

In chapter 5, we extend the proposed decentralized coded caching technique to wireless ad hoc networks. We will prove the significant capacity gain of this approach in wireless ad hoc networks for reactive routing scenarios compared to uncoded caching and further we prove that this coding technique is very efficient in providing security in wireless ad hoc networks. We will show that when the number of cached files is very large this technique satisfies in the perfect secrecy conditions proposed by Shannon in [88] and therefore this technique is capable of achieving asymptotic perfect secrecy in wireless ad hoc networks. This provides an information theoretically secure solution for caching proprietary contents in wireless networks which is immune to attackers in time as opposed to computationally security which may be broken with time. We also compute the cache hit probability and we will show that this solution results in a much

higher cache hit probability compared to uncoded caching. We will also propose a secure caching update algorithm in the end of this chapter.

In chapter 6, we extend the idea of RLF coding to storage systems. These codes due to their intrinsic randomness can provide significant security gains. Hence, they could be potentially very useful in sensitive cloud storage application which should be information theoreticly secure and immune to attackers in time. Also, in many distributed storage applications like Peer-to-Peer (P2P) distributed storage systems or distributed storage systems in which some of the servers are under the control of an oppressive government, a user wants to download a content from a pool of distributed servers in a way that the servers cannot determine which content is requested by the user. This is widely known as Private Information Retrieval (PIR) problem. We will introduce a new PIR scheme based on random query vectors and we will show that this PIR technique is robust against many colluding servers. These random queries are designed in a way that they can be used to retrieve any desired content while prevent any malicious agent with the knowledge of up to half of the random queries to gain information about the requested content. This is an important feature of the proposed technique that provides privacy in the presence of many colluding servers. Such a feature has not been presented in prior information theoretic PIR approaches for coded storage systems. Our proposed Secure And Private Information Retrieval (SAPIR) scheme provides both security and privacy for information retrieval in storage and cloud systems.

Chapter 2

Index Coding Based Caching in Information-Centric Networks

In this chapter we introduce a new method of caching based on *index coding* for Information Centric Networks (ICN). The index coding problem relates to transmission policies when the source node broadcasts encoded data to users with side information. In this chapter we extend the index coding problem to cases when the source node can reach users through multihop communications. This approach is called Modified Index Coding (MIC) which can be applied to both wireless and wired networks. We demonstrate the benefits of our approach by applying MIC to Information-Centric Networks (ICN). We show that the combination of ICN and MIC requires a hybrid caching scheme that includes both central and distributed caching to support two different goals. The approach results in a combination of conventional caching in ICN and a new distributed caching scheme across nodes in the network. Our analysis demonstrates that capacity

improvement can be achieved by the new architecture. Simulation results compare the capacity improvement to traditional ICN architecture. The results studied in this chapter are published in [51].

2.1 Motivation and related works

Index Coding (IC) [9, 10] focuses on wireless architectures in which the transmitter utilizes coding in order to take advantage of broadcast nature of the wireless channel. For example in satellite communications, combination of coding at the transmitter side with caching at the receivers achieves higher capacity gains compared to traditional schemes. The original IC problem was based on the assumption that the channel is a broadcast channel and the source can reach all the nodes in one transmission. We will modify this concept to accommodate other classes of channels such as wired or multihop wireless networks. Figure 2.1 demonstrates an example of IC problem with source having six messages and each node N_1 to N_6 needs one message and has a subset of all messages. For example, node N_4 needs message m_4 while it has prior side information m_5 and m_6. The objective is to find an optimal encoding scheme that allows all nodes to receive their required messages with minimum number of transmissions. In this figure, the source node can either broadcast to all the nodes (IC problem) or communicate with nodes using multihop communications (MIC problem). If all the nodes in figure 2.1 are within transmission range of the source, then the source can send only two encoded messages of $m_1 + m_2 + m_3$ and $m_4 + m_5 + m_6$ to allow all nodes to

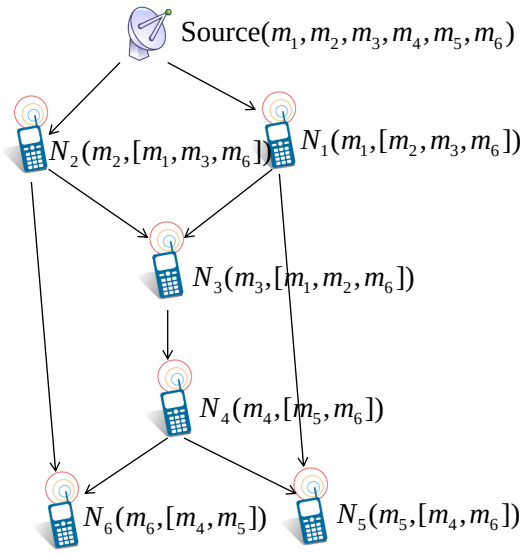

Figure 2.1: An example of Multihop Index Coding (MIC) problem.

retrieve their requested messages instead of six transmissions. For example, node N_4 can add m_5 and m_6 to encoded message $m_4 + m_5 + m_6$ in order to recover m_4.

Information Centric Network (ICN) architectures [1, 2, 5, 6, 81, 83] were introduced based on the premise that in most Internet applications, users are interested in accessing the content regardless of the location of delivery as long as the information is secure. ICN focuses on the content delivery without any consideration on where the content is obtained. The key question that ICN attempts to address is how to securely deliver huge amount of contents that are distributed in different locations and requested by many users. ICN addresses this question by utilizing a naming architecture where the

content is retrieved by its name and defining a naming taxonomy that makes the content independent of its source or location. Further, it allows the contents to be cached in the network, preferably close to the destinations. This unique content recovery requires content-based routing in order to find the content in the network using appropriate name resolution infrastructure to map a name to one copy of the content. This new approach has provided significant benefits for content delivery in the network at the expense of additional overhead to keep track of content locations in different caches. A natural question that we intend to address is that "how can we take advantage of caching in ICN and modify IC concept in order to increase the overall throughput capacity of the network?"

In the following parts of this chapter, in Section 2.2, we focus on defining the modified index coding (MIC) and the new hybrid caching schemes. Section 2.3 describes the proposed ICN architecture. In section 2.4, we prove that MIC can be efficiently utilized to increase the capacity of the ICN. Simulation results in section 2.5 demonstrate the throughput capacity improvement by combining MIC with ICN.

2.2 Model and problem formulation

2.2.1 Modified Index Coding (MIC)

The IC problem [9,10] was originally formulated for wireless broadcast channels such as satellite communication applications. We assume the source node can access the receiver nodes through multihop communications. The new scheme, called Modified

Index Coding (MIC) [85] technique, can be applied to different types of networks such as wireless multihop communication network, wired network or a hybrid network consisting of both wired and wireless channels. For example, in a corporate environment, we can have information transported in a wired medium while the last hop is a wireless access point with nodes connected to the infrastructure through a wireless modem. Such scenarios are becoming more common in medium and large size corporate campuses.

Definition 2.2.1. *A Modified Index Code* $\mathbb{MIC}(\mathbb{M}, \mathbb{N})$ *consists of a set of* k *messages* $\mathbb{M} = \{m_1, \ldots, m_k\}$ *and a set of receiver nodes* $\mathbb{N}$*. Each node* N_i *stores a subset of messages called* $\mathbb{L}_i \subseteq \mathbb{M}$ *and requests one message*[1] m_i *that node* N_i *does not have, i.e.,* $m_i \in M$ *and* $\mathbb{L}_i \subseteq \mathbb{M} \setminus \{m_i\}$*. Each message can be divided into* n *packets, i.e.,* $m_i = \{m_{i1}, \ldots, m_{in}\}$*. Each packet also belongs to an alphabet taken from a* q*-ary finite Field* Γ*. Therefore, we have* $m_i = \{m_{i1}, \ldots, m_{in}\} \in \Gamma^n$*. We further define* $\aleph = \{m_{11}, \ldots, m_{1n}, \ldots, m_{k1}, \ldots, m_{kn}\} \in \Gamma^{nk}$*. At any given time,* $k_1 \leq k$ *receiver nodes are requesting some messages. The source defines groups of receiver nodes* $k_1^1, \ldots, k_1^p$*, where* $k_1^1 + \ldots + k_1^p = k_1$*. Modified index code for* $\mathbb{MIC}(\mathbb{M}, \mathbb{N})$ *is a function* $f_i : \Gamma^{nk} \rightarrow \Gamma^{\ell_i}$*, for an integer value of* ℓ_i *and groups of nodes* k_1^i *that satisfies for each receiver node* $N_i = (m_i, \mathbb{L}_i) \in \mathbb{N}$ *within this group* k_1^i *a function* $\Phi_i : \Gamma^{\ell_i + n|\mathbb{L}_i|} \rightarrow \Gamma^n$ *such that the desired message can be decoded for that particular node, i.e.,* $\Phi_i(f_i(\aleph), \mathbb{L}_i) = m_i, \forall \aleph \in \Gamma^{nk}$*.*

Note that in this new definition, we multicast different coded messages to these p groups, each one consists of k_1^i receiver nodes where $1 \leq i \leq p$. IC definition subsumes

[1] Extension of this definition to multiple requests is straightforward.

MIC since broadcast is a special case of multicast when the set of receiver nodes include the entire network. Under the new definition, we can apply MIC for both wireless and wired networks. MIC is utilized to design a new ICN architecture.

Figure 2.1 can be used for an example of MIC. Solid lines represent one hop and dotted lines represent multiple hops. Suppose that any two nodes can communicate only when there is an arrow between them. For instance, the source node in Figure 2.1 can only communicate with nodes N_1 and N_2. In this example, $p = 2$ and $k_1^1 = \{N_1, N_2, N_3\}$ and $k_1^2 = \{N_4, N_5, N_6\}$. The source node multicasts two encoded messages of $m_1 + m_2 + m_3$ and $m_4 + m_5 + m_6$ to k_1^1 and k_1^2 groups respectively. These two encoded messages are the minimum number of transmissions (optimum) that will achieve the desired outcome. However, in general the problem of finding the best encoding strategy is an NP-hard problem.

2.2.2 Hybrid Caching

One of the main features of ICN is the ability of the network to cache the requested contents in the network at different locations in order to serve the users with lower latency and improve the throughput capacity in the process by bringing the contents closer to the users. This feature seems to be very attractive in separating the content from any unique source node in order to find the nearest content to the client node. However, this by itself creates certain challenges for network designers. One major challenge is how to locate the closest cached content in the network? Another challenge is to design a caching policy that will increase the throughput capacity while reducing

the latency. We introduce a new hybrid caching technique that requires minimum overhead related to locating stored cache contents.

In our architecture, we use caching for two different purposes. We cache the contents in some locations in the network in order to provide it to users similar to the original approach in ICN. However, most users have significant storage capacity that is not used. For example, it is now common for a laptop to have Terabyte of storage not utilized by majority of the users. We use this enormous distributed storage capacity for improving the data distribution in ICN architecture. We propose that each user shall keep any data object that is requesting from the network. Therefore, each user allocates a predefined portion of its storage to keep the data objects that it has already obtained. By the discussion that we had in the previous section related to MIC, it should be clear that the data stored by different users throughout the network will be used to extract the desired message when the node receives a combination of multiple messages. The encoded message is multicasted by the local cache system that is serving these nodes. The contents that are cached by the nodes will not be used for transmission to other requesting nodes unlike current ICN architectures. The problem with distributed caching is the significant overhead associated to this approach. We suggest that the requested contents by different users should always come from the local cache system or from source node that has the content. In our proposed architecture, caching are used for two different purposes as described above.

Note that by taking advantage of the MIC, we need one multicast session to replace k_1^i unicast sessions. It is easy to see [100] that one multicast session always

consumes less channel bandwidth in the network than k_1^i unicast sessions. This reduction in network resource usage can be very helpful specially when the size of contents are large. Another advantage of this architecture is the fact that since the local router that caches the contents knows what contents each node has stored before, there is no need to update each cache (node) information in the network. As long as the local router can keep track of the cached contents in different nodes, then the overhead is very small. Note that since each node receives the requested content via this local router, then that information is already available to the local router.

2.3 Proposed ICN Architecture

In this section, we will describe how to take advantage of the MIC concept in order to derive a new architecture for ICN. We assume each group of nodes in the network is served by a unique router that also caches the information. In this context, if a node requests a content, this request will be directed toward that particular router. The router either sends the information directly to the node or finds the source for the requested content. Further, we assume when a node receives a content, it will keep this content in its cache. In Figure 2.2, all the routers that are shaded are responsible for serving different groups of nodes. The selection of these routers is based on the number of nodes that are connected to that router either directly or through multiple hops. In general, there is a trade-off between latency, speed of router and the maximum number of nodes assigned to a router.

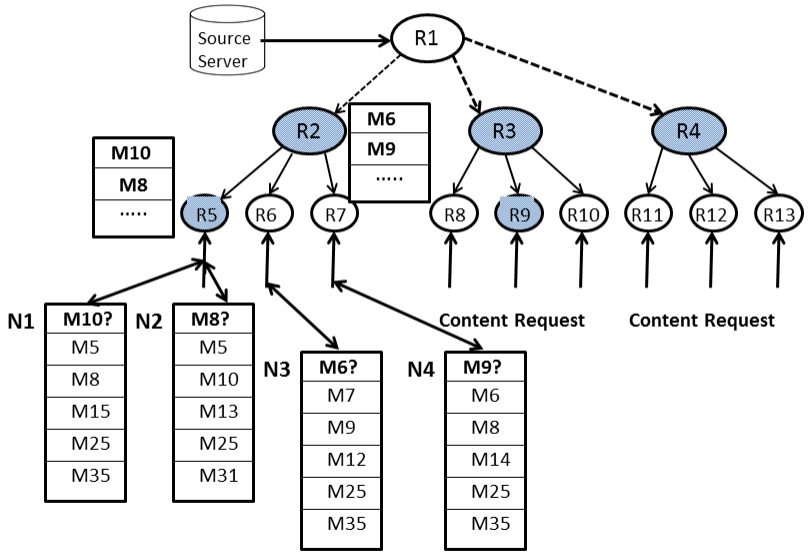

Figure 2.2: Example of application of MIC in ICN.

When a content is delivered to a node, the node will keep a copy of this content in its cache. Now let's assume each node has a subset of contents in its cache (see figure 2.2). When some of these nodes request different[2] contents from the local router, this router network encodes [104] the requested contents by utilizing modified index coding technique to minimize the total number of transmissions to serve all these nodes. The optimum encoding selection is an NP-hard problem. We will introduce some sub-optimal approach for the encoding scheme.

If the content is not available in the serving router, then the router will request the content from the source (or another router on the way toward the source). Once it receives the content, it will encode the received content along with other requested contents and multicast it to the requesting nodes. This router also keeps one copy of the content in its cache. As we can see, under the new architecture, we do not use an aggressive caching approach that each router or node caches the contents but rather a subset of the routers cache the contents. The assumption here is that most of the contents that are being requested by a node in each group of nodes, will likely be requested by another node in that group. This is particularly true since most of the content request popularity are heavy-tailed and have a distribution close to Zipf distribution. Prior studies [8,101] have shown that multiple layer caching or cooperative caching does not provide significant improvement for Zipfian distribution. Recent study [35] has suggested caching scheme that takes advantage of this distribution and caches at the edges of the network. Our approach has some similarities to the technique proposed

[2]Some nodes can request the same content.

in [35] by suggesting that it is sufficient to cache in a subset of routers at the edges of the network.

Figure 2.2 demonstrates an example for our proposed ICN architecture. Router R_5 serves nodes N_1 and N_2 and router R_2 serves nodes N_3 and N_4. Let nodes N_1 and N_2 request messages m_{10} and m_8 respectively. Both these nodes are served by router R_5. When these nodes send request to this router, the router will multicast $m_{10} + m_8$ to these two nodes. Node N_1 can add the received encoded message with m_8 to obtain m_{10} and node N_2 can similarly obtain m_8. Nodes N_3 and N_4 are served by router R_2 via routers R_6 and R_7 respectively. R_2 multicasts $m_6 + m_9$ and each node can retrieve its requested data. All these operations are carried in Galois Field. It is quite possible that more complicated combinations of messages are sent by routers in order for nodes to decode their requested messages.

As long as the caching policies of the users are known by the router, the router knows for each user which contents are being stored and which contents are evicted after the user reaches its maximum caching capacity. This clearly requires additional processing power for each router that is involved in caching but it also reduces the overhead. There exists another overhead associated to a node requesting a content. Since each user has an assigned local router to serve that user, the request is always directed toward that router. Clearly, the overhead associated to finding the content in the network by the local router is similar to the current network architectures. Therefore, our proposed architecture simplifies the overhead and content routing challenges in ICN systems.

2.4 ICN Capcity Improvement using MIC

In this section, we study the problem of capacity improvement in the proposed combination of ICN architecture and MIC. We assume the content popularity follows a Zipfian-like distribution which is supported by many studies [15, 18]. MIC provides additional capacity gain when a subset of contents have higher popularity among nodes such as in Zipfian-like distribution.

In the remainder of this section, we assume that the network is a hybrid network with the last hop is between a wireless router (like 802.11) and mobile users. Similar to many studies on index coding (IC) [10, 21, 32] that demonstrate dependency graph is a useful tool for analysis of these networks, we take advantage of this concept in this paper.

Definition 2.4.1. (Dependency Graph): *Given an instance of index coding problem, the dependency graph $G(V, E)$ is defined as follows:*

- *Each client N_i corresponds to a vertex in V, $N_i \in V$.*
- *There is a directed edge in E from N_i to N_j if and only if the client N_i is requesting a content that is already cached in N_j.*

It is known from [21] that if we choose the right encoding vectors, for any vertex disjoint cycle in the dependency graph we can save at least one transmission. Therefore, the number of vertex-disjoint cycles in the dependency graph can serve as a lower bound for the number of saved transmissions in any IC problem. The same result

also holds for an MIC problem since MIC is similar to the IC problem and a dependency graph can be defined for each subnetwork.

Assume that we have an ICN system that is utilizing MIC. Let's assume the set of contents available in the entire network as $M = \{m_1, m_2, ..., m_k\}$ with m_1 being the most popular content and m_k being the least popular content in the network[3]. Also, assume that the users $N = \{N_1, N_2, ..., N_l\}$ are being served by a specific router R and user N_i is requesting content with popularity index r_i in the current time instant. For the sake of simplicity of calculations, let's assume that each user has a cache of fixed size δ in which, the contents with indices $C_i = \{c_{i1}, ..., c_{i\delta}\}$ are stored.

As suggested by [14,15,18], we can assume a Zipfian distribution with parameter s for content popularity distribution in the network. This means that the probability that N_i requests any content with popularity index r_i at any time instant is

$$\Pr[N_i \text{ requests content with index } r_i] = \frac{{r_i}^{-s}}{H_{k,s}}, \tag{2.1}$$

where $H_{k,s}$ is the kth generalized harmonic number with parameter s defined as

$$H_{k,s} = \sum_{j=1}^{k} \frac{1}{j^s}. \tag{2.2}$$

Lemma 2.4.2. *When $s > 1$, for every $0 < \epsilon < 1$, there exists an integer $h = \Theta(1)$ with respect to k such that for every i,*

$$Pr[r_i \leq h] \geq 1 - \epsilon. \tag{2.3}$$

[3]Popularity decreases with index number.

Proof. Based on the Zipfian distribution assumption, this probability is equal to

$$\Pr[r_i \leq h] = \frac{H_{h,s}}{H_{n,s}}. \tag{2.4}$$

If $s > 1$, we have $H_{n,s} < H_{\infty,s} = \zeta(s)$ where $\zeta(.)$ denotes the Reimann Zeta function. If we choose h to be the first integer such that $H_{h,s} \geq \zeta(s)(1-\epsilon)$, we are guaranteed to have $\Pr[r_i \leq h] \geq 1-\epsilon$. Notice that h can be chosen independently of n, i.e., $h = \Theta(1)$. □

Remark 2.4.3. If $0 \leq s \leq 1$, to make sure that $\Pr[r_i \leq h] \geq 1-\epsilon$, the value of h should grow with n but the growth rate is so slow that we can still treat h as a constant number with respect to n and use Lemma 2.4.2 for practical purposes.

Therefore, based on Lemma 2.4.2 and Remark 2.4.3, if h is chosen a sufficiently large integer, with probability close to one, all users are requesting contents with maximum popularity index h. Before going further, we will bring up the following Lemma from [33].

Lemma 2.4.4. Let $d_c > 1$ and $l \geq 24d_c$, then any graph $G^l_{f(l,d_c)}$ with l nodes and at least $f(l,d_c) = (2d_c-1)l - 2d_c^2 + d_c$ edges contains d_c disjoint cycles or $2d_c - 1$ vertices of degree $l-1$ (its structure is then uniquely determined). [4]

Proof. The proof is in [33]. □

The fact that there are strong correlation between cached contents and new requests due to Zipfian distribution of contents, it is clear that MIC will provide some

[4] Clearly, the theorem is valid when the number of edges is more than $f(l,d_c)$.

capacity improvement. We will now demonstrate the efficiency of applying MIC to ICN in the following theorem.

Theorem 2.4.5. *For large values of h and l, using MIC in ICN can save on average $\Omega(lp_0)$ transmissions for any router serving l nodes in any time slot where*

$$p_0 = \frac{h^{-s}}{H_{k,s}}. \tag{2.5}$$

Proof. The dependency graph $G(V, E)$ in our problem is composed of l vertices $N_1, N_2, ..., N_l$ which corresponds to the l nodes served by a local router. Note that the existence of an edge in dependency graph depends on the probability that a node is requesting a content that another node has already cached. Therefore, this is a non-deterministic graph with some probability for the existence of each edge between any two vertices. In this non-deterministic dependency graph, the probability of existence of edge (N_i, N_j) in E is equal to the probability that the content r_i requested by N_i is cached in node N_j. If we assume these two probabilities are independent, then the probability of a directed edge (N_i, N_j) in E is at least

$$\Pr[(N_i, N_j) \in E] \geq \frac{r_i^{-s}}{H_{k,s}}. \tag{2.6}$$

Using lemma 2.4.2 and remark 2.4.3, when the value of h is large enough, the probability that r_i is less than h is very close to one. This means that with a probability close to one, the edge presence probability in equation (2.6) can be lower bounded by p_0. Therefore, the maximum number of vertex-disjoint cycles in $G(V, E)$ can be lower bounded by the maximum number of vertex-disjoint cycles in an Erdős-Réyni random graph $G'(l, p_0)$

with l nodes and edge presence probability p_0. Now we can use lemma 2.4.4 to find a lower bound on the number of vertex disjoint cycles in $G'(l, p_0)$. This in turn will give us a lower bound on the number of vertex-disjoint cycles in $G(V, E)$.

Notice that $G'(l, p_0)$ is an Erdős-Réyni random graph and it can have a maximum of $l(l-1)$ directed edges. However, since every edge in this graph exists with a probability of p_0, the expected value of the number of edges is $l(l-1)p_0$. This means that if d_c is chosen to be an integer such that

$$l(l-1)p_0 \geq (2d_c - 1)l - 2d_c^2 + d_c, \tag{2.7}$$

then on average, $G'(l, p_0)$ will have d_c disjoint cycles. We can easily verify that $d_c = \frac{lp_0}{2}$ satisfies (2.7). Therefore, with a probability close to one (for large enough values of h), the dependency graph $G(V, E)$, on average has at least $\Omega(lp_0)$ vertex disjoint cycles. This can be directly applied to prove the theorem. □

Theorem 2.4.6. *The acheived lower bound in theorem 2.4.5 is a tight order bound of* $\Theta(l)$.

Proof. Notice that the maximum number of vertex-disjoint cycles in any graph with l vertices is $\frac{l}{2}$. However, theorem 2.4.5 proves that the maximum number of vertex-disjoint cycles in our graph is lower bounded by $\Omega(lp_0)$. This suggests that this is indeed a tight order bound. □

We can further prove that many properties of the dependency graph are independent of the number of contents in the network. This implies that the properties

of the dependency graph are mainly dominated by the most popular contents in the network. As an example of these properties, we can consider the problem of finding a clique of size k_1 in the dependency graph. A clique of size k_1 in the dependency graph has an interesting interpretation since such a clique means that there exist a set of k_1 users $N_b = \{N_{b_1}, N_{b_2}, ..., N_{b_{k_1}}\}$ such that for every $1 \leq i \leq k_1$ and every $1 \leq j \leq k_1$ when $j \neq i$, we have $r_{b_i} \in C_{b_j}$. This means that a simple linear index code $\sum_{i=1}^{k_1} m_{b_i}$ can be used by the local router to send the content m_{b_i} to user N_{b_i} for all $1 \leq i \leq k_1$ in just one transmission. Each user will then be able to decode the requested message using its cached contents. The following theorem proves that the existence of a clique of size k_1 is independent of the total number of contents in the network k and only depends on the popularity index s.

Theorem 2.4.7. Let's define the content index requested by node i in j^{th} cache space as $c_{i,j} \in C_i$. We assume the content request probability follows a Zipfian distribution and the users request independent contents in different time slots[5]. Then the probability of finding a set of k_1 users $N_b = \{N_{b_1}, N_{b_2}, ..., N_{b_{k_1}}\} \subseteq \mathbb{N}$ for which a single index code can be used to transmit the requested content m_{b_i} with index number r_{b_i} to N_{b_i} for $1 \leq i \leq k_1$ is independent of the total number of contents in the network.

Proof. The probability that a specific set of users $\{N_{b_1}, N_{b_2}, ..., N_{b_{k_1}}\}$ form a clique of size k_1 is given by

$$P_{b_1,b_2,...,b_{k_1}} = \Pr[r_{b_i} \in C_{b_j} \text{ for } 1 \leq \forall i, j \leq k_1, j \neq i\ ], \tag{2.8}$$

[5]A user will not request a content that it has already cached.

where C_{b_j} is the set representing the contents that node N_{b_j} caches. Assuming that the users are requesting contents independently of each other, this probability can be simplified as

$$P_{b_1,b_2,\ldots,b_{k_1}} = \prod_{i=1}^{k_1} \prod_{j=1,j\neq i}^{k_1} \Pr[r_{b_i} \in C_{b_j}]. \tag{2.9}$$

Since the contents that are requested by each user in different time slots are independent of each other, in the steady state when the caches of all nodes are filled we have

$$\Pr[r_{b_i} \in C_{b_j}] = 1 - \Pr[r_{b_i} \notin C_{b_j}] = 1 - \prod_{u=1}^{\delta} \Pr[r_{b_i} \neq c_{b_j,u}]. \tag{2.10}$$

Notice that we assume a Zipfian distribution for the content request in the ICN network, therefore, $\Pr[r_{b_i} = c_{b_j,u}]$ will be equal to the probability that user N_{b_j} requests content with index r_{b_i} in u^{th} cache space. Hence

$$\Pr[r_{b_i} = c_{b_j,u}] = \frac{{r_{b_i}}^{-s}}{H_{n,s}} \tag{2.11}$$

Therefore, combining equations (2.11) and (2.10), we arrive at

$$\Pr[r_{b_i} \in C_{b_j}] = 1 - \left(1 - \frac{{r_{b_i}}^{-s}}{H_{n,s}}\right)^{\delta}. \tag{2.12}$$

Equation (2.9) can be simplified as

$$P_{b_1,b_2,\ldots,b_{k_1}} = \prod_{i=1}^{k_1} \left(1 - \left(1 - \frac{{r_{b_i}}^{-s}}{H_{n,s}}\right)^{\delta}\right)^{k_1-1}. \tag{2.13}$$

In order to have a clique of size k_1, we need to include the possibility over all groups of k_1 users. Therefore, the probability of having a clique of size k_1 (which we denote by P) should be summed up over all of these choices.

$$P = \sum_{b_1,b_2,\ldots,b_{k_1} \subseteq \mathbb{N}} P_{b_1,b_2,\ldots,b_{k_1}}$$

$$= \sum_{b_1,b_2,...,b_{k_1}\subseteq\mathbb{N}} \prod_{i=1}^{k_1} \left(1-\left(1-\frac{{r_{b_i}}^{-s}}{H_{n,s}}\right)^{\delta}\right)^{k_1-1} \tag{2.14}$$

Note that for any $0 \leq x \leq 1$ and positive integer δ we have

$$1-(1-x)^{\delta} \geq x. \tag{2.15}$$

Hence,

$$\begin{aligned} P &\geq \sum_{b_1,b_2,...,b_{k_1}\subseteq\mathbb{N}} \prod_{i=1}^{k_1} \left(\frac{{r_{b_i}}^{-s}}{H_{n,s}}\right)^{k_1-1}, \\ &= \frac{\sum_{b_1,b_2,...,b_{k_1}\subseteq\mathbb{N}} \prod_{i=1}^{k_1} r_{b_i}^{-s(k_1-1)}}{H_{n,s}^{k_1(k_1-1)}}. \end{aligned} \tag{2.16}$$

In order to simplify this expression, we use the *elementary symmetric polynomial* notation. If we have a vector $V_l = (v_1, v_2, ..., v_l)$ of length l, then the k_1-th degree elementary symmetric polynomial of these variables is denoted as

$$\sigma_{k_1}(V_l) = \sigma_{k_1}(v_1, ..., v_l) = \sum_{1\leq i_1<i_2<..<i_{k_1}\leq l} v_{i_1}...v_{i_{k_1}}. \tag{2.17}$$

Using this notation and by defining $Y_l = (r_1^{-s(k-1)}, r_2^{-s(k-1)}, ..., r_l^{-s(k-1)})$, we can write

$$P \geq \frac{\sigma_{k_1}(Y_l)}{H_{n,s}^{k_1(k_1-1)}}. \tag{2.18}$$

Notice that since the content request probability follows a Zipfian distribution, we have

$$\Pr[r_j \leq h] = \frac{H_{h,s}}{H_{n,s}}. \tag{2.19}$$

Therefore, for a specific group of users $N_{b_1}, N_{b_2}, ..., N_{b_{k_1}}$, the probability that they all request contents from the top h most popular contents is given by

$$\Pr[r_{b_1} \leq h, ..., r_{b_{k_1}} \leq h] = \prod_{j=1}^{k_1} \Pr[r_{b_j} \leq h] = \left(\frac{H_{h,s}}{H_{n,s}}\right)^{k_1}. \tag{2.20}$$

Using lemma 2.4.2, we can verify that for values of $h = \Theta(1)$ with respect to n, the ratio $\frac{H_{h,s}}{H_{n,s}}$ can be very close to one. This, along with the fact that l can possibly be much larger than k_1, means that with a very high probability, there exists for each set of users $\{N_{b_1}, N_{b_2}, ..., N_{b_{k_1}}\}$ that the requests come only from the h most popular contents. This implies that with a very high probability, $\sigma_{k_1}(Y_l) \geq h^{-k_1 s(k_1-1)}$. Also, notice that

$$H_{n,s} < H_{\infty,s} = \zeta(s) < \infty. \tag{2.21}$$

Therefore, the lower bound of (2.18) is obtained as

$$P \geq \left(\frac{h^{-s}}{\zeta(s)}\right)^{k_1(k_1-1)}. \tag{2.22}$$

This lower bound does not depend on the total number of contents in the network (k), and only depends on l, h, s and k_1. The results means that regardless of the amount of contents in the network, there is always a constant lower bound for the probability of finding a clique of size k_1. This implies that MIC can actually be very practical in large networks. The result also indicates that the probability of a clique tends exponentially to zero as k_1 increases. In practice, it is usually sufficient to look for cliques of sizes 2, 3, and 4. □

2.5 Simulations

In this section, we demonstrate the performance of the new architecture compared to the original ICN without using MIC. Figure 2.3 shows the probability that a specific content that the users are requesting be available in the endpoint router. In

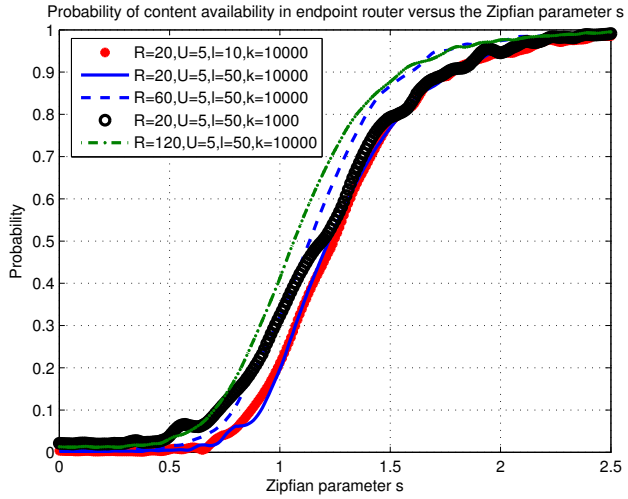

Figure 2.3: Probability of requesting a content that is already available in the edge router cache.

this figure, R denotes the size of local router cache, U is the size of user's cache, l is the number of users served by the router and k is the total number of contents in the network. As this plot suggests, the probability of content availability in the endpoint router approaches one as the Zipfian parameter is increased. Notice that this probability goes to one regardless of the number of available contents in the network, number of users and other factors. However, figure 2.3 suggests that this probability is slightly higher when the number of contents is smaller and/or the router has a larger cache size. The fact that many requested contents have been already cached in the router implies that some nodes also store them in their caches. Therefore, we predict that the introduction of MIC to ICN architecture will be very useful. For these simulations, R denotes the size of endpoint router cache and U denotes the size of user's cache.

Figure 2.4, shows the simulation results for four different sets of parameters. In

this figure, we have plotted the effective average number of packets sent per transmission. The baseline is ICN with no MIC which is equivalent of one content per transmission. In this figure, R denotes the size of local router cache, U is the size of user's cache, l is the number of users served by the router and k is the total number of contents in the network. Note that one transmission can contain more than one package as it may serve multiple users. We have assumed that the users are requesting contents based on a Poisson distribution. In each time slot, the local router updates the dependency graph based on the received requests from users and also removes some of the edges for the contents that it has transmitted. The algorithm is a simple heuristic approach that at each time slot, the local router first searches for a clique in the dependency graph. If we can find a clique of size k_1 in the dependency graph, we can save $k_1 - 1$ transmissions by transmitting a simple index code to all the users in the clique. If there is no clique in the dependency graph, then we will search for cycles. As mentioned before, each cycle can save one transmission per cycle. Note that this is a simple heuristic suboptimal approach and finding the best solution is beyond the scope of this paper. In fact, to find the actual benefit that we can achieve by using MIC, we need to find the optimal rate for the index coding problem. Note that finding a clique of maximum size in the dependency graph or the optimal index coding rate is an NP-hard problem. For our simulation, we have searched for cliques and cycles of maximum size four. Even with this simple algorithm, we were able to show that the MIC can close to double the average number of packets per transmission in each time slot for certain values of the Zipfian parameter. Clearly, the optimal MIC rate is larger than the results obtained by our

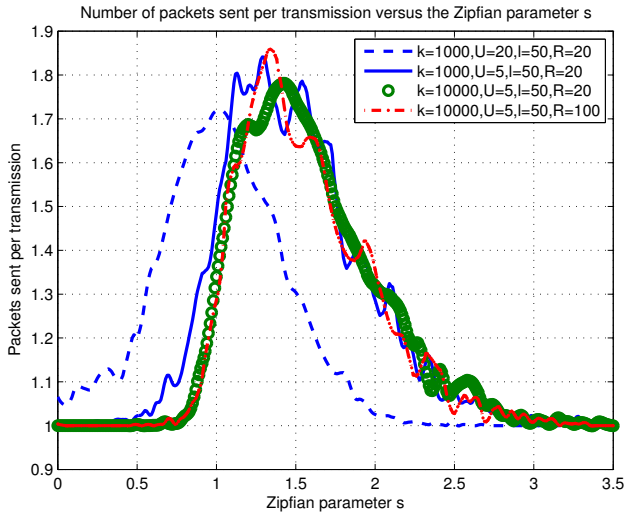

Figure 2.4: Average number of packets transmitted in each time slot when using MIC.

simple algorithm. Note that the users are using Least Recently Used (LRU) caching policy for eviction of overflow contents.

When s is a small value, then the distribution of content request is close to uniform distribution. Under this condition, the dependency graph is very sparse because there is a small probability that a user requests a content that is already cached by another user. Clearly, there is no benefit for using MIC in this case. Similarly, when s is a very large number, most users are asking for the same content and therefore, the router broadcasts the content to all of them which is equivalent of average of one content per transmission. The main benefit of MIC happens for medium values of s between 0.5 and 2 which is usually the case in practical networks. Note that an ICN with no MIC, will have always one content per transmission which is the baseline.

Chapter 3

Index Coding Based Caching in Cellular Networks

In this chapter we study the benefits of using index coding for caching in the next generation of cellular networks which are deploying wireless distributed femtocaching infrastructure proposed by Golrezaei et. al. By taking advantage of multihop communications in each cell, the number of required *femtocaching helpers* is significantly reduced. This reduction is achieved by using the underutilized storage and communication capabilities in the User Terminals (UTs), which results in reducing the deployment costs of distributed femtocaches.

A multihop index coding technique is proposed to code the cached contents in helpers to achieve order optimal capacity gains. As an example, we consider a wireless cellular system in which contents have a popularity distribution and demonstrate that our approach can replace many unicast communications with multicast communication.

We will prove that simple heuristic linear index code algorithms based on graph coloring can achieve order optimal capacity under Zipfian content popularity distribution. The results studied in this chapter are published in [53].

3.1 Motivation

With the recent pervasive surge in using wireless devices for video and high speed data transfer, it seems eminent that the current wireless cellular networks cannot be a robust solution to the ever-increasing wireless bandwidth utilization demand. Researchers have been recently focused on laying down the fundamental grounds for future cellular networks to overcome such problems.

Deploying home size base stations is proposed as a solution in [20]. The solution in [20] is based on the idea of femtocells in which many small cells are deployed throughout the network to cover the entire network. Deploying many small cells in the network with reliable backhaul links requires significant capital investment. Therefore, Golrezaei et. al. [37] proposed femtocaching as an alterante solution to overcome this problem. In this approach, in every cell along with the main base station, smaller base stations with low-bandwidth backhaul links and high storage capabilities are deployed to create a wireless distributed caching infrastructure. These small base stations which are called caching helpers (or simply helpers), will store popular contents in their caches and use their caches to serve *User Terminal (UT)* requests. Therefore, in networks with high content reuse, the backhaul utilization will be significantly reduced using this ap-

proach. If the requested content is not available in the helper's cache, UTs can still download the content from their low-bandwidth backhaul links to the base station. The proposed technique in [37] requires deployment of large number of femtocaches in order to cover all the nodes in the network.

On the other hand, it is well-known that web content request popularity follows Zipfian-like distributions [15]. This content popularity distribution implies that few popular contents are widely requested by the network UTs. We assume UTs store their requested contents and therefore, helpers can multicast multiple requests by taking advantage of coding to reduce the total number of transmissions.

In this chapter, we propose to use index coding to code the contents in helpers before transmission. As mentioned before, index coding is a source coding technique proposed in [7] which takes advantage of UTs' side information in broadcast channels to minimize the required number of transmissions. In index coding, the source (e.g. base station) designs codes based on the side information stored in requesting nodes. The coded information is broadcasted to the UTs that use the information together with their cached contents to decode the desired content. Index coding improves bandwidth utilization by minimizing the number of required transmissions. We propose to extend index coding approach from broadcast one-hop communication to multihop scenarios which will be explained in details.

Our main motivation to use index coding is the high storage availability in UTs to improve the achievable throughput of the future wireless cellular networks. Current improvements in high density storage systems has made it possible to have personal

devices with Terabytes of storage capability. This ever-increasing trend provides future personal wireless devices with huge under-utilized storage capabilities. Future wireless devices can use their storage capability to store the contents that they have already requested. In an index coding setting, many UTs that are requesting different contents can receive a coded file which is multicasted to them and then each UT uses the information in its cache to decode its requested content from the received coded file. There is an important equivalence between index coding and network coding as stated in [31,32] and therefore, the results in this chapter can also be stated based on a network coding terminology.

We will prove that index coding can be efficiently used to encode the contents by helpers under a Zipfian distribution model. The encoded contents can be relayed through multiple hops to all the UTs being served by that helper.

The optimal index coding solution is an NP-Hard problem [64]. However, we will show that even using linear index codes can result in order optimal capacity gains in these networks. We believe that this coding technique can serve as a complement to the solution proposed in [37]. As clearly articulated in [72], in any caching problem we are faced with two phases of cache placement and cache delivery. While [37] proposes efficient cache placement algorithms, we will be focusing on efficient delivery methods for their solution. We will show that the problem of delivery in [37] can be efficiently addressed by using index coding in the helpers.

Recent discussions on standards for future 5G cellular networks are focused on providing high bandwidth for Device-to-Device communications (D2D). Examples of

such approaches are the IEEE 802.11ad standard (up to 60GHz [4]) and the millimeter-wave proposal which can potentially enable up to 300GHz of bandwidth for D2D communications [3, 12]. This potential abundant D2D bandwidth can be utilized to relay the coded contents inside an ad hoc network which is being served by a helper. It is indeed such excessive storage and bandwidth capabilities of future wireless systems that make our solution feasible.

Deploying many femtocaching helpers is not economically efficent. On the other hand, since UTs have significant D2D capabilities, they can efficiently participate in content delivery through multihop D2D communications. In our proposal, we suggest to deploy few helpers which can deliver the contents to neighboring UTs through multihop D2D communications. This can reduce the network deployment and maintenance costs.

The rest of this chapter is organized as follows. Section 3.2 reviews the related works and section 3.3 describes the proposed network model that is similar to [37] with the addition of using multihop communications and index coding. In section 3.4, we will explain the scaling laws of capacity improvement using index coding and relaying. Section 3.5 demonstrates that index coding algorithms can achieve order optimal gains. Section 3.6 describes the simulation results.

3.2 Related work

The problem of caching when a server is transmitting contents to clients was studied in [72] from an information theoretic point of view. The authors introduced two phases of *cache placement* and *content delivery*. For a femtocaching solution, efficient cache placement algorithms are proposed in [37]. In this chapter, we focus on efficient content delivery algorithms through index coding and multihop D2D communications for a femtocaching solution.

There has been significant research on index coding since it was proposed in [7]. The practical implementation of index coding for wireless applications was proposed in [21] by proposing cycle counting methods. A dynamic index coding solution for wireless broadcast channels is proposed in [78]. In [78], a wireless broadcast station is considered and a simple set of codes based on cycles in the dependency graph is provided. They show the optimality of these codes for a class of broadcast relay problems. In this chapter, we prove that codes based on cycles can acheive order optimal capacity gains in networks with Zipfian content request distribution.

Approximating index coding solution is proved [7] to be an NP-Hard problem. However, efficient heuristics has been proposed in [22] which are based on well-known graph coloring algorithms. Other references like [32] and [31] have studied the connections and equivalence between index coding and network coding. Tran et al. [93] studied a single hop wireless link from a network coding approach and showed similar results to index coding. Ji et al. [47] studied theoretical limits of caching in D2D communication

networks.

Study of coding techniques in networks with high content reuse has recently attracted the attention of researchers. Montpetit et al. [74] studied the applications of network coding in Information-Centric Networks (ICN). Wu et al. [102] studied network coding in Content Centric Networks (CCN) which is an implementation of ICN. Leong et al. [66] proposed a linear programming formulation to deliver contents optimally in today's IP based Content Delivery Networks (CDN) using network coding. Llorca et at. [68] proposed a network-coded caching-aided multicasting technique for efficient content delivery in CDNs. This paper extends Index Coding to multihop communication and shows that linear codes can achieve order optimal capacity gains in such networks.

3.3 Network Model

We assume a network model for future wireless cellular networks in which femtocaching helpers with significant storage capabilities are deployed throughout the network to assist the efficient delivery of contents to UTs through reliable D2D communications. Such helpers are characterized with low rate backhaul links which can be wired or wireless. They will also have localized, high-bandwidth communication capabilities. With their significant storage capacity, in a network with high content reuse, many of the content requests for the UTs inside a cell can be satisfied directly by the helpers.

Each helper is serving a wireless ad-hoc network in which the UTs are utilizing

high bandwidth D2D communication techniques such as millimeter wave and IEEE 802.11ad technologies. This high bandwidth D2D communication enables the UTs to relay data from a nearby helper to all the UTs that are within transmission range. We assume that the path lifetime between the helper and UTs is longer than the time required to transmit the content. For large files and when UTs are moving fast, one solution is to divide the content into smaller files and treat each file separately. We assume that the network is connected which can be justified by the large number of UTs that will be available in the future wireless cellular networks.

Further, we take advantage of index coding and the side information cached in UTs to significantly reduce the required number of helpers and consequently, reduce the infrastructure maintenance and deployment costs. To demonstrate the effectiveness of using multihop communications, we consider similar assumptions as [37] with a macro base station placed in the center of a cell with radius 400 meters serving 1000 UTs and a transmission range of 100 meters [3] for D2D communication. As shown in Figure 3.1, with only 4 helpers uniformly located in the cell, 100% and 80% of nodes are covered with 3 and 2 hop communications respectively. Covering all the UTs in the same cell with only one hop communication requires up to 27 helpers [37]. The simulation in Figure 3.1 is carried over a cell with radius 400 meters and with a communication range of 100 meters. Clearly, for a vehicle or a mobile UT operating in the cell, the handover probability will be significantly reduced if the number of helpers shrinks from 27 to 4.

We assume that n UTs denoted by $\mathbb{N} = \{N_1, N_2, ..., N_n\}$ are being served by a helper. There are m contents $\mathbb{M} = \{M_1, M_2, ..., M_m\}$ available with M_1 as the most

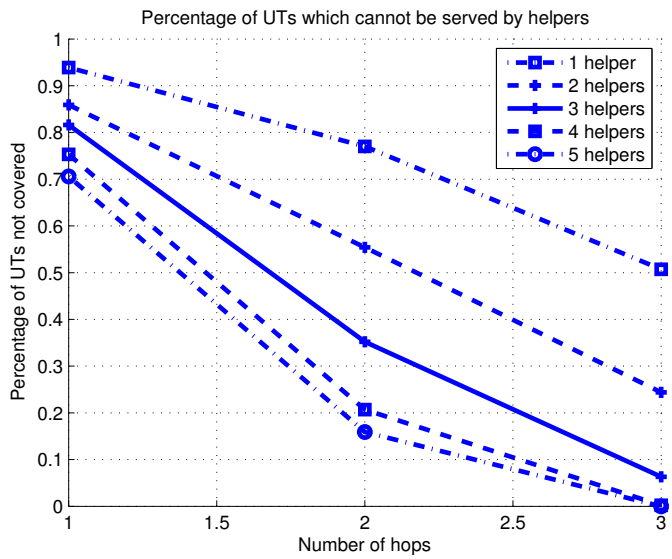

Figure 3.1: Pertentage of UTs not covered versus the maximum number of hops traveled.

popular content and M_m as the least popular content in the network . Let's assume UT N_i requests a content with popularity index r_i in the current time interval. Each UT has a cache of fixed size δ in which contents with indices $C_i = \{c_{i1}, ..., c_{i\delta}\}$ are stored. Therefore, we assume that UT N_i caches contents $M_{c_{i1}}, M_{c_{i2}}, \ldots, M_{c_{i\delta}}$ and the set of cached content indices in N_i is represented by C_i. Therefore, if we denote the set of cached contents in UT N_i by $\mathcal{M}_i$, then we have $\mathcal{M}_i = \{M_j \mid j \in C_i\}$. The requested content index is shown by r_i.

Let's assume that we have n UTs each with a set of side information $\mathcal{M}_i$. The formal definition of index code described in [7] is given below.

Definition 3.3.1. An *index code* on a set of n UTs each with a side information set $\mathcal{M}_i \subseteq \mathbb{M}$, and a requesting content $M_{r_i} \in \mathbb{M}$ for $i = 1, 2, \ldots, n$ is defined as a set of codewords in $\{0, 1\}^l$ toghether with

1. An encoding function $\mathcal{E}$ mapping inputs in $\{0,1\}^m$ to codewords and

2. A set of decoding functions $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_n$ such that $\mathcal{D}_i(\mathcal{E}(\mathbb{M}), \mathcal{M}_i) = M_{r_i}$ for $i = 1, 2, \ldots, n$.

In the above definition, the length of the index code l denotes the number of required transmissions to satisfy all the content requests of the n UTs. The encoding function $\mathcal{E}$ is applied to the contents by the helper and the decoding functions $\mathcal{D}_i$s are applied individually by UTs to decode their desired contents from the encoded content using their cached information.

Figure 3.2 shows a helper H serving 6 UTs N_1, N_2, N_3, N_4, N_5, and N_6. Each arrow represents a link with high bandwidth D2D communication capability. Let's assume UTs N_1, N_2 and N_5 request contents M_3, M_1 and M_4 while storing $\{M_1, M_4\}$, $\{M_3, M_4\}$ and $\{M_1, M_3\}$ respectively. Using index coding requires 3 channel usages while without index coding, we need 5 channel usages. This is true since using index coding, the helper creates the XOR combination of contents M_1, M_3 and M_4 as $M_1 \oplus M_3 \oplus M_4$ and broadcasts this coded content to its neighboring UTs. Either of N_1 and N_2 can immediately reconstruct their requested content from this coded content. For instance, N_1 which is requesting M_3 can decode its requested content by using it's cached information and XOR operation on the encoded message to retrieve the requested content M_3, i.e., $(M_1 \oplus M_3 \oplus M_4) \oplus M_1 \oplus M_4 = M_3$. After N_2 receives $M_1 \oplus M_3 \oplus M_4$, it relays it to node N_4 which simply broadcasts it to N_5. UT N_5 will again use XOR operation to decode it's desired content M_4 by adding it's cached contents to the coded

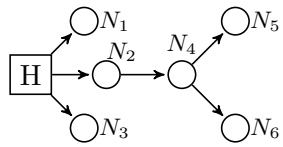

Figure 3.2: Example of a wireless multihop network being served by the helper H.

content that it has received, i.e., $(M_1 \oplus M_3 \oplus M_4) \oplus M_1 \oplus M_3 = M_4$. Therefore, only 3 transmissions are needed.

To satisfy the content requests through multihop D2D without index coding, 5 transmissions are needed as the helper should transmit M_3 to N_1, M_1 to N_2, M_4 to N_2, and N_2 relays M_4 to N_4 and N_4 relays it to N_5.

We assume a Zipfian distribution with parameter $s > 1$ for content popularity distribution in the network. This means that the probability that UT N_i requests any content with index r_i at any time instant is given by

$$\Pr[N_i \text{ requests content with index } r_i] = \frac{r_i^{-s}}{H_{m,s}}, \tag{3.1}$$

where $H_{m,s} = \sum_{j=1}^{m} \frac{1}{j^s}$ denotes the mth generalized harmonic number with parameter s.

Remark 3.3.2. In this chapter we only focus on Zipfian content request probability with parameter $s > 1$ which is a *non-heavy-tailed* probability distribution. Our results are correct for any type of non-heavy-tailed probability distribution and the extension to heavy-tailed probability distribution is the subject of future work.

Dependency graph is a useful analytical tool [7, 21, 32] that is widely used in index coding literature. A formal definiton of dependency graph is brought in 2.4.1 in

chapter 2. Notice that the dependency graph does not represent the actual physical links between UTs in the network. This is a virtual graph in which each edge represents the connection between a UT that is requesting a content and a UT that caches this content. As discussed in [21, 78], every cycle in the dependency graph is representative of a connection between UTs and it can save one transmission. For every clique in the dependency graph, all the requesting UTs in the clique can be satisfied by a simple linear XOR index code. The complement of dependency graph is called *conflict graph*. This graph is of significant interest since any clique in the dependency graph gives rise to an independent set[1] in the conflict graph. Therefore, well-known graph coloring algorithms over conflict graph can be used to find simple linear XOR index codes. The dependency and conflict graphs in our network are random directed graphs. In the next section, we will use the properties of these graphs to find the capacity gains and propose simple index coding solutions.

To prove our results, we have used Least Recently Used (LRU) or Least Frequently Used (LFU) cache policies. Similar results can be produced for other caching policies. LRU caching policy assumes that most recently requested contents are kept in the cache. In LRU caching, in each time slot the content that is requested in the previous time slot is stored in the first cache location. If this content was already available in the cache, it is moved from that location to the first cache location. If the content was not available, then the content that is least recently used is discarded and the most recently requested content is cached in the first cache location. Any other content is

[1] An independent set is a set of vertices in a graph for which none of the vertices are connected by an edge.

relocated to a new cache location such that the contents appear in the order that they have been requested.

In LFU caching policy, the contents are cached based on their request frequency. Highly popular contents are stored in the first locations of the cache and contents with lower request frequency are cached in the bottom locations of the cache.

Computing the probability of having content with index r_i in cache C_j, $\Pr[r_i \in C_j]$, turns out to be complicated for LRU or LFU caching policies. A simple lower bound on this probability for LRU can be found by noticing that $\Pr[r_i \in C_j]$ is greater than the probability that UT N_j have requested the content with index r_i in the most recent time slot and therefore it is located at the top of the cache. This lower bound can be derived using equation (3.1).

$$\begin{aligned}\Pr[r_i \in C_j] &= \Pr[c_{j1} = r_i] + \sum_{l=2}^{\delta} \Pr[c_{jl} = r_i] \\ &\geq \Pr[c_{j1} = r_i] = \frac{r_i^{-s}}{H_{m,s}}. \end{aligned} \tag{3.2}$$

To prove that the same lower bound holds for LFU, notice that $\Pr[r_i \in C_j]$ is larger than the same probability when the cache size is $M = 1$. Therefore,

$$\Pr[r_i \in C_j] \geq \Pr[r_i \in C_j \mid M = 1] = \Pr[c_{j1} = r_i] = \frac{r_i^{-s}}{H_{m,s}}. \tag{3.3}$$

In the subsequent sections, we will use these lower bounds to prove our results.

We will state our results in terms of order bounds. To avoid any confusion, we use the following order notations [59]. We denote $f(n) = \mathrm{O}(g(n))$ if there exist $c > 0$ and $n_0 > 0$ such that $f(n) \leq cg(n)$ for all $n \geq n_0$, $f(n) = \Omega(g(n))$ if $g(n) = \mathrm{O}(f(n))$, and $f(n) = \Theta(g(n))$ if $f(n) = \mathrm{O}(g(n))$ and $g(n) = \mathrm{O}(f(n))$.

3.4 Order optimal capacity gain

In this section, we will prove that index coding can significantly decrease the number of transmissions in the network. We will specifically use the Zipfian content distribution in the underlying content distribution network. To do so, we will first state and prove the following lemma.

Lemma 3.4.1. Let's consider a Zipfian content distribution with parameter $s > 1$ and parameter $h_\epsilon = \epsilon^{\frac{1}{1-s}}$ where $0 < \epsilon < 1$. For every i with popularity index r_i that is less than h_ϵ, the request probability is at least $1 - \epsilon$.

Proof. Based on the Zipfian distribution assumption and equation (3.1), the probability that the requested content has a popularity of at most h_ϵ is equal to

$$\Pr[r_j \leq h_\epsilon] = \frac{H_{h_\epsilon,s}}{H_{m,s}}. \tag{3.4}$$

In order to satisfy $\Pr[r_j \leq h_\epsilon] \geq 1 - \epsilon$, we should have

$$\begin{aligned} \epsilon \geq 1 - \Pr[r_j \leq h_\epsilon] = 1 - \frac{H_{h_\epsilon,s}}{H_{m,s}} &= \frac{1}{H_{m,s}} \sum_{j=h_\epsilon+1}^{m} j^{-s} \\ = \frac{1}{H_{m,s}} \sum_{i=1}^{\lfloor \frac{m}{h_\epsilon} \rfloor - 1} \sum_{j=ih_\epsilon+1}^{(i+1)h_\epsilon} j^{-s} &+ \frac{1}{H_{m,s}} \sum_{j=\lfloor \frac{m}{h_\epsilon} \rfloor h_\epsilon+1}^{m} j^{-s}. \end{aligned} \tag{3.5}$$

If we have

$$\epsilon \geq \frac{1}{H_{m,s}} \sum_{i=1}^{\lfloor \frac{m}{h_\epsilon} \rfloor} \sum_{j=ih_\epsilon+1}^{(i+1)h_\epsilon} j^{-s}, \tag{3.6}$$

then the inequality in (3.5) will certainly hold since the right hand side in (3.6) is larger than the right hand side in (3.5). Note that, for $ih_\epsilon + 1 \leq j \leq (i+1)h_\epsilon$ we have

$j^{-s} < (ih_\epsilon)^{-s}$. Hence,

$$\sum_{j=ih_\epsilon+1}^{(i+1)h_\epsilon} j^{-s} < \sum_{j=ih_\epsilon+1}^{(i+1)h_\epsilon} (ih_\epsilon)^{-s} = h_\epsilon (ih_\epsilon)^{-s} = i^{-s} h_\epsilon^{1-s} \tag{3.7}$$

This means that if

$$\epsilon \geq h_\epsilon^{1-s} \frac{1}{H_{m,s}} \sum_{i=1}^{\lfloor \frac{m}{h_\epsilon} \rfloor} i^{-s}, \tag{3.8}$$

then (3.6) holds since the right hand side of inequality (3.6) is smaller than the right hand side of inequality (3.8) as shown by (3.7). Notice that since,

$$\frac{1}{H_{m,s}} \sum_{i=1}^{\lfloor \frac{m}{h_\epsilon} \rfloor} i^{-s} = \frac{\sum_{i=1}^{\lfloor \frac{m}{h_\epsilon} \rfloor} i^{-s}}{\sum_{i=1}^{\lfloor \frac{m}{h_\epsilon} \rfloor} i^{-s} + \sum_{\lfloor \frac{m}{h_\epsilon} \rfloor + 1}^{m} i^{-s}} < 1,$$

if h_ϵ is chosen such that

$$\epsilon \geq h_\epsilon^{1-s}, \tag{3.9}$$

then all of the inequalities in equations (3.5), (3.6), (3.7) and (3.8) will be valid and hence $\Pr[r_j \leq h_\epsilon] \geq 1 - \epsilon$. Therefore, in order to have $\Pr[r_j \leq h_\epsilon] \geq 1 - \epsilon$, it is enough to choose h_ϵ such that (3.9) is valid. Hence, if h_ϵ is chosen to be at least equal to

$$h_\epsilon = \epsilon^{\frac{1}{1-s}}, \tag{3.10}$$

then we have $\Pr[r_j \leq h_\epsilon] \geq 1 - \epsilon$. Notice that the choice of h_ϵ in (3.10) is such that it only depends on ϵ and is independent of m. □

For instance for $s = 2$ and $\epsilon = 0.01$, h_ϵ can be chosen as 100 (regardless of the size of m). This implies that for a Zipfian distribution with $s = 2$, 100 highly popular contents among any large number of contents would account for 99% of the total content

requests. Therefore, if h_ϵ is chosen as in equation (3.10), with a probability of at least $1-\epsilon$ all content requests have popularity index of at most h_ϵ. Now define p_ϵ as

$$p_\epsilon \triangleq \frac{h_\epsilon^{-s}}{H_{m,s}} = \frac{\epsilon^{-\frac{s}{1-s}}}{H_{m,s}}. \tag{3.11}$$

Based on above discussion, in our instance of index coding dependency graph, with a probability of at least $1-\epsilon$ edges are present with a probability of at least p_ϵ. We will discuss this in more details later in the proof for Theorem 3.4.2. Notice that for large values of m, p_ϵ is also independent of m since in that case $H_{m,s} \approx \zeta(s)$ and p_ϵ only depends on ϵ and s.

As stated in [21], if we choose the right encoding vectors for any index coding problem, for any vertex disjoint cycle in the dependency graph we can save one transmission. Therefore, the number of vertex-disjoint cycles[2] in the dependency graph can serve as a lower bound for the number of saved transmissions in any index coding problem. Number of vertex disjoint cycles is also used in [78] as a way of finding the lower bound for index coding gain. To count the number of vertex-disjoint cycles in our random dependency graph, we will use Lemma 2.4.4 in chapter 2 which is originally proved as Theorem 1 in [33].

Note that the dependency graph is a directed graph and in order to use Lemma 2.4.4, we need to construct an undirected graph. Let's denote the directed and undirected random graphs on n vertices and edge presence probability p_ϵ by $\overrightarrow{G}(n, p_\epsilon)$ and $G(n, p_\epsilon)$, respectively. In a directed graph $\overrightarrow{G}(n, p_\epsilon)$, the probability that two vertices are connected by two opposite directed edges is p_ϵ^2. Therefore, we can build an undirected

[2]These are the cylces that do not have any common vertex.

graph $G(n, p_\epsilon^2)$ with the same number of vertices and an edge between two vertices if there are two opposite directed edges in the directed graph $\overrightarrow{G}(n, p_\epsilon)$ between these two UTs. Hence, $\overrightarrow{G}(n, p_\epsilon)$ essentially contains a copy of $G(n, p_\epsilon^2)$. Note that there are some edges between UTs in $\overrightarrow{G}(n, p_\epsilon)$ that do not appear in $G(n, p_\epsilon^2)$. This fact was also observed in [41]. Therefore, a lower bound on the number of disjoint cycles for $G(n, p_\epsilon^2)$ implies a lower bound on the number of disjoint cycles for $\overrightarrow{G}(n, p_\epsilon)$.

In the following theorems, we will use Lemma 2.4.4 to prove that using index coding to code the contents can be very efficient.

Theorem 3.4.2. Assume all UTs are utilizing LRU or LFU caching policies for a Zipfian content request distribution with parameter $s > 1$. Index coding can save $\Omega(np_\epsilon^2)$ transmissions for any helper serving n UTs with a probability of at least $1 - \epsilon$ for any $0 < \epsilon < 1$.

Proof. Consider a Zipfian distribution with parameter $s > 1$ and let $0 < \epsilon < 1$ be fixed. The dependency graph $\overrightarrow{G}(V, E)$ in our problem is composed of n vertices $N_1, N_2, \ldots, N_n$ which correspond to the n UTs that are served by a helper. Note that the existence of an edge in dependency graph depends on the probability that a UT is requesting a content and another UT has already cached that content[3]. Therefore, this is a non-deterministic graph with some probability for the existence of each edge between the two vertices. In this non-deterministic dependency graph, the probability of existence of edge (N_i, N_j) in E is equal to the probability that content r_i requested by N_i, is already cached in N_j. Therefore, with LRU or LFU caching policy assumption and using equations (3.2)

[3]This edge has no relationship with the actual physical link between two UTs.

and (3.3), we arrive at

$$\Pr[(N_i, N_j) \in E] = \Pr[r_i \in C_j] \geq \frac{r_i^{-s}}{H_{m,s}}. \tag{3.12}$$

Using Lemma 3.4.1 for any $0 < \epsilon < 1$, if h_ϵ is chosen as $h_\epsilon = \epsilon^{\frac{1}{1-s}}$, then with a probability of at least $1-\epsilon$, any requested content has a popularity index r_i less than h_ϵ. This means that with a probability of at least $1-\epsilon$, the edge presence probability in equation (3.12) can be lower bounded by p_ϵ. Therefore, with a probability of at least $1-\epsilon$, maximum number of vertex-disjoint cycles in our directed dependency graph $\overrightarrow{G}(V, E)$ can be lower bounded by the maximum number of vertex-disjoint cycles in an Erdős-Réyni random graph $\overrightarrow{G}(n, p_\epsilon)$ with n vertices and edge presence probability p_ϵ. Now we can use Lemma 2.4.4 and undirected graph $G(n, p_\epsilon^2)$ to find a lower bound on the number of vertex disjoint cycles in $\overrightarrow{G}(n, p_\epsilon)$. This in turn, will give us a lower bound on the number of vertex-disjoint cycles in $\overrightarrow{G}(V, E)$.

Note that $G(n, p_\epsilon^2)$ is an undirected Erdős-Réyni random graph on n vertices and edge presence probability p_ϵ^2. This graph has a maximum of $n(n-1)$ undirected edges. However, since every undirected edge in this graph exists with a probability of p_ϵ^2, the expected value of the number of edges in graph $G(n, p_\epsilon^2)$ is $n(n-1)p_\epsilon^2$. This means that if d in Lemma 2.4.4 with $v = n$ is chosen to be an integer such that

$$n(n-1)p_\epsilon^2 \geq (2d-1)n - 2d^2 + d, \tag{3.13}$$

then on average, $G(n, p_\epsilon^2)$ will have either d disjoint cycles or $2d-1$ vertices of degree $n-1$. For the purpose of our paper we can easily verify that for large enough values of n, $d^\star = \lfloor \frac{np_\epsilon^2}{24} \rfloor$ satisfies equation (3.13) (Notice that the condition $24d^\star \leq n$ in Lemma

2.4.4 is also met). Therefore based on Lemma 2.4.4, the graph $G(n, p_\epsilon^2)$ either has at least $d^\star = \lfloor \frac{np_\epsilon^2}{24} \rfloor$ disjoint cycles or $2d^\star - 1 = 2\lfloor \frac{np_\epsilon^2}{24} \rfloor - 1$ vertices with degree $n-1$. As mentioned before, $\overrightarrow{G}(n, p_\epsilon)$ essentially contains a copy of $G(n, p_\epsilon^2)$. Consequently, $\overrightarrow{G}(n, p_\epsilon)$ either has at least $d^\star = \lfloor \frac{np_\epsilon^2}{24} \rfloor$ disjoint cycles or $2d^\star - 1 = 2\lfloor \frac{np_\epsilon^2}{24} \rfloor - 1$ vertices with degree $n-1$. The number of vertices in graph $\overrightarrow{G}(n, p_\epsilon)$ is n. Therefore, the latter case gives rise to a situation where there are $2d^\star - 1 = 2\lfloor \frac{np_\epsilon^2}{24} \rfloor - 1$ vertices which are connected to any other vertex in $\overrightarrow{G}(n, p_\epsilon)$ through undirected edges. This condition results in having a clique of size $2\lfloor \frac{np_\epsilon^2}{24} \rfloor - 1$ in $\overrightarrow{G}(n, p_\epsilon)$.

In summary, $\overrightarrow{G}(n, p_\epsilon)$ has either $d^\star$ disjoint cycles or it contains a clique of size $2d^\star - 1$. Hence, with a probability of at least $1 - \epsilon$, the dependency graph $\overrightarrow{G}(V, E)$ on average has either $d^\star$ disjoint cycles or it contains a clique of size $2d^\star - 1$. In either of these cases $d^\star = \lfloor \frac{np_\epsilon^2}{24} \rfloor$ transmissions can be saved using index coding. This proves the theorem. □

Theorem 3.4.3. Index coding through cycle counting and clique partitioning can save $\Theta(n)$ transmissions in a network with n UTs and Zipfian content request distribution with parameter $s > 1$.

Proof. In Theorem 3.4.2, we proved that in a network with Zipfian content request distribution and for a fixed $0 < \epsilon < 1$, with a probability of at least $1 - \epsilon$, the index coding dependency graph either has $d^\star = \lfloor \frac{np_\epsilon^2}{24} \rfloor$ disjoint cycles or it contains a clique of size $2d^\star - 1 = 2\lfloor \frac{np_\epsilon^2}{24} \rfloor - 1$. Consider the following situations,

1. The dependency graph has $d^\star = \lfloor \frac{np_\epsilon^2}{24} \rfloor$ disjoint cycles. In this case, for a fixed ϵ,

p_ϵ is a constant which does not depend on n. Hence, cycle counting can result in at least $d^\star = \lfloor \frac{np_\epsilon^2}{24} \rfloor = \Omega(n)$ transmission savings. This is a lower bound on the number of transmission savings.

On the other hand, notice that the maximum number of vertex-disjoint cycles in any graph with n vertices cannot be greater than $\frac{n}{2}$ as shown in Figure 3.3. Therefore, the maximum number of transmission savings using cycle counting is $\frac{n}{2} = O(n)$. This is an upper bound on the number of saved transmissions. Since the order of upper and lower bounds are the same, it can be concluded that the number of saved transmissions scales as $\Theta(n)$.

2. The dependency graph contains a clique of size $\mathrm{k}^\star = 2d^\star - 1 = 2\lfloor \frac{np_\epsilon^2}{24} \rfloor - 1$. Through clique partitioning we will be able to save at least $\mathrm{k}^\star - 1$ transmissions by sending only one transmission. Hence, through clique partitioning, we will be able to save at least $\mathrm{k}^\star - 1 = \Omega(\frac{np_\epsilon^2}{12}) = \Omega(n)$ transmissions by sending only one transmission to the UTs forming that specific clique. This is a lower bound on the number of saved transmissions.

   On the other hand, if the dependency graph is a perfectly complete graph on n nodes which means that every requested content is available in all other UTs' caches, then all the requested transmissions can be satisfied by one transmission which is a linear XOR combination of all requested contents.

   Hence, the number of transmission savings is equal to $n - 1 = O(n)$. Notice that this is the maximum number of transmission savings since we at least need

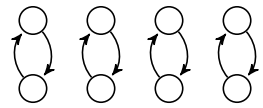

Figure 3.3: Example of a dependency graph on $n = 8$ nodes with maximum possible number of vertex disjoint cycles $n/2 = 4$.

1 transmission to satisfy all content requests. This means that the number of transmission savings is upper bounded by $\mathrm{O}(n)$. Since the upper and lower order bounds are the same, we conclude that the transmission saving scales as $\Theta(n)$.

□

We can further prove that many properties of the dependency graph are independent of the total number of contents and only depends on the most popular contents in the network. As an example of these properties, we can consider the problem of finding a clique of size k in the dependency graph. A clique of size k in the dependency graph has an interesting interpretation since all the requests in this clique can be satisfied with one multicast transmission. The following theorem proves that the probability of existence of a clique of size k is lower bounded by a value which is independent of the total number of contents in the network, m, and only depends on the popularity index s.

Theorem 3.4.4. If LRU or LFU caching policy is used and the content request probability is Zipfian distribution, then the probability of finding a set of k UTs $N_b = \{N_{b_1}, N_{b_2}, ..., N_{b_k}\} \subseteq \mathbb{N}$ for which a single linear index code (XOR operation) can be used to transmit the requested content r_{b_i} to N_{b_i} for $1 \leq i \leq k$ can be lower bounded

by a value that with a probability close to one is independent of the total number of contents in the network.

Proof. The probability that a specific set of UTs $\{N_{b_1}, N_{b_2}, ..., N_{b_k}\}$ form a clique of size k is

$$P_{b_1,b_2,...,b_k} = \Pr[r_{b_i} \in C_{b_j} \text{ for } 1 \leq \forall i, j \leq k, j \neq i\ ]. \tag{3.14}$$

Assuming that the UTs are requesting contents independently of each other, this probability can be simplified as

$$P_{b_1,b_2,...,b_k} = \prod_{i=1}^{k} \prod_{j=1, j\neq i}^{k} \Pr[r_{b_i} \in C_{b_j}]. \tag{3.15}$$

Using equations (3.2) and (3.3), we arrive at

$$\Pr[r_{b_i} \in C_{b_j}] \geq \frac{{r_{b_i}}^{-s}}{H_{m,s}}. \tag{3.16}$$

Equation (3.15) can be lower bounded as

$$P_{b_1,b_2,...,b_k} \geq \prod_{i=1}^{k} \left(\frac{{r_{b_i}}^{-s}}{H_{m,s}}\right)^{k-1}. \tag{3.17}$$

The probability to have a clique of size k is computed by considering all $\binom{n}{k}$ groups of k UTs. Hence, the probability of having a clique of size k denoted by P_k is given by

$$\begin{aligned} P_k &= \sum_{b_1,b_2,...,b_k \subseteq N} P_{b_1,b_2,...,b_k} \geq \sum_{b_1,b_2,...,b_k \subseteq N} \prod_{i=1}^{k} \left(\frac{{r_{b_i}}^{-s}}{H_{m,s}}\right)^{k-1} \\ &= \frac{\sum_{b_1,b_2,...,b_k \subseteq N} \prod_{i=1}^{k} {r_{b_i}}^{-s(k-1)}}{H_{m,s}^{k-1}}. \end{aligned} \tag{3.18}$$

In order to simplify this expression, we use the *elementary symmetric polynomial* notation. If we have a vector $V_n = (v_1, v_2, ..., v_n)$ of length n, then the k-th degree elementary

symmetric polynomial of these variables is denoted as

$$\sigma_k(V_n) = \sigma_k(v_1, ..., v_n) = \sum_{1 \leq i_1 < i_2 < .. < i_k \leq n} v_{i_1} ... v_{i_k}. \tag{3.19}$$

Using this notation and by defining $Y_n \triangleq (r_1^{-s(k-1)}, r_2^{-s(k-1)}, ..., r_n^{-s(k-1)})$, we have $P_k \geq \frac{\sigma_k(Y_n)}{H_{m,s}^{k-1}}$. Since the content request probability follows a Zipfian distribution, we have $\Pr[r_j \leq h_\epsilon] = \frac{H_{h_\epsilon,s}}{H_{m,s}}$. Therefore, for a specific group of UTs $N_{b_1}, N_{b_2}, ..., N_{b_k}$, the probability that they all request contents from the top h_ϵ most popular contents is given by

$$\Pr[r_{b_1} \leq h, ..., r_{b_k} \leq h_\epsilon] = \prod_{j=1}^{k} \Pr[r_{b_j} \leq h_\epsilon] = \left(\frac{H_{h_\epsilon,s}}{H_{m,s}}\right)^k. \tag{3.20}$$

We have already proved in Lemma 3.4.1 that for large values of m and $h_\epsilon = \epsilon^{\frac{1}{1-s}}$, the ratio $\frac{H_{h_\epsilon,s}}{H_{m,s}}$ is greater than $1 - \epsilon$. Besides this, the fact that n is most likely much larger than k, means that with a very high probability, for each set of UTs $\{N_{b_1}, N_{b_2}, ..., N_{b_k}\}$, the requests come only from the h_ϵ most popular contents. This implies that with a high probability, $\sigma_k(Y_n) \geq \binom{n}{k} h_\epsilon^{-ks(k-1)}$. Also, notice that $H_{m,s} < \zeta(s) < \infty$. Therefore, with a probability close to one, P_k can be lower bounded as

$$P_k \geq \left(\frac{h_\epsilon^{-ks}}{\zeta(s)}\right)^{k-1}. \tag{3.21}$$

This lower bound does not depend on m and only depends on n, ϵ, s and k. □

Theorem 3.4.4 states that regardless of the number of contents in the network, there is always a constant lower bound for the probability of finding a clique of size k. The result hints the potential use of linear index coding in these networks. In the next

section, we will prove that linear index coding can indeed be very useful and can be used to construct codes acheiving order optimal capacity gains.

Remark 3.4.5. The above capacity improvement is found for a traditional single hop index coding scenario. For our proposed multihop setup, similar gains still hold. In our proposed setting, we consider communications for a small number of hops and therefore multihop communication can only affect the capacity gain by a constant factor and the order bound results will not be affected.

3.5 Heuristics acheiving order optimal capacity

Both optimal and approximate solutions [7, 64] for the general index coding problem are NP-hard problems. Some efficient heuristic algorithms for the index coding problem were proposed [22] which can provide near optimal solutions. In some of these heuristic algorithms, the authors reduce the index coding problem to the graph coloring problem.

Notice that every clique in the dependency graph of a specific index coding problem, can be satisfied with only one transmission which is a linear combination of all contents requested by the UTs corresponding to the clique. Therefore, solving the clique partitioning problem, which is the problem of finding a clique cover of minimum size for a graph [36], yields a simple linear index coding solution. The minimum number of cliques required to cover a graph can be regarded as an upper bound on the minimum number of index codes required to satisfy the UTs. Index coding rate is defined as

the minimum number of required index codes to satisfy all the UTs. Since lower index coding rates translate into higher values of transmission savings (or index coding gains)[4] as discussed in [21], the number of transmission savings found in the clique partitioning problem is in fact a lower bound on the total number of transmission savings found from the optimal index coding scheme (or the optimal index coding gain).

On the other hand, solving the clique partitioning problem for any graph $G(V, E)$ is equivalent to solving the graph coloring problem for the complement graph $\bar{G}(V, \bar{E})$ which is a graph on the same set of vertices V but containing only the edges that are not present in E. This is true because every clique in the dependency graph, gives rise to an independent set in the complement graph. Therefore, if we have a clique partitioning of size χ in the dependency graph, we have χ distinct independent sets in the complement graph. In other words, the chromatic number of the complement graph is χ.

The above argument allows us to use the rich literature on the chromatic number of graphs to study the index coding problem. In fact, any graph coloring algorithm running over the conflict graph can be directly used to obtain an achievable index coding rate. If running such an algorithm over the conflict graph results in a coloring of size χ, this coloring gives rise to a clique cover of size χ in the dependency graph and an index coding of rate χ with index coding gain of $n - \chi$ which is a lower bound for the total number of transmission savings using the optimal index code[5].

[4] In a dependency graph of n UTs with the index coding rate of χ, the number of saved transmissions, $n - \chi$, is called the index coding gain.

[5] Notice that since the optimal index coding rate is upper bounded by the size of the minimum clique cover (which is equal to the chromatic number of the conflict graph), the value of transmission savings

Therefore, considering the chromatic number of the conflict graph, we can find a lower bound on the asymptotic index coding gain. To do so, we use the following theorem from [13],

Theorem 3.5.1. For a fixed probability p, $0 < p < 1$, almost every random graph $G(n, p)$ (a graph with n UTs and the edge presence probability of p) has chromatic number,

$$\chi_{G(n,p)} = -\left(\frac{1}{2} + o(1)\right) \log(1-p) \frac{n}{\log n} \tag{3.22}$$

We will now use Theorem 3.5.1 and the designed undirected graph $G(n, p_\epsilon^2)$ to find the number of transmission savings using a graph coloring based heuristic in our network.

Theorem 3.5.2. Using a graph coloring algorithm, in a network with n UTs almost surely gives us a linear index code with gain

$$l = \Theta\left(n + \left(\frac{1}{2} + o(1)\right) \frac{n}{\log n} \log p_\epsilon^2\right). \tag{3.23}$$

Proof. Assume that a helper is serving n UTs where n is a large number. As discussed in Theorem 3.5.1, the index coding gain is lower bounded by $n - \chi$ where χ is the chromatic number of the conflict graph. However, notice that on average the chromatic number of our non-deterministic conflict graph is upper bounded by the chromatic number of an

that we can achieve using the optimal index code is lower bounded by $n - \chi$.

undirected random graph with edge existence probability of $1-p_\epsilon^2$. To prove this, notice that in the dependency graph, the probability of edge existence between two vertices is at least p_ϵ which implies that the number of edges in the dependency graph is on average greater than or equal to the number of edges in a directed Erdos-Reyni random graph $\overrightarrow{G}(n, p_\epsilon)$. However, we know that the number of edges in $\overrightarrow{G}(n, p_\epsilon)$ is at least equal to the number of edges in an undirected Erdos-Reyni random graph $G(n, p_\epsilon^2)$. Therefore, the conflict graph which is the complement of dependency graph, on average has less edges compared to a random graph with edge existence probability of $1 - p_\epsilon^2$ and consequently, its chromatic number cannot be greater than the chromatic number of $G(n, 1-p_\epsilon^2)$. Given these facts, the index coding gain is lower bounded by $n-\chi_{G(n,1-p_\epsilon^2)}$. Since $1 - p_\epsilon^2$ is fixed, Theorem 3.5.1 implies that the chromatic number of the conflict graph is equal to

$$\chi_{G(n,1-p_\epsilon^2)} = -\left(\frac{1}{2} + o(1)\right) \log p_\epsilon^2 \, \frac{n}{\log n} \tag{3.24}$$

This proves that the index coding gain is lower bounded by $\Omega(n\times\{1+\left(\frac{1}{2} + o(1)\right) \frac{1}{\log n} \log p_\epsilon^2\})$ which asymptotically tends to n. However, the maximum index coding gain of n UTs is also n. Therefore, this coding gain is also a tight bound. □

Remark 3.5.3. Theorem 3.5.2 presents the index coding gain using a graph coloring algorithm which only counts the number of cliques in the dependency graph. The gain in Theorem 3.4.2 counts the number of disjoint cycles in the dependency graph. Theorem 3.4.3 proves that index coding gain in Theorem 3.4.2 is $\Theta(n)$ which means that it is order optimal. Theorem 3.5.2 is also proving the same result. Therefore, a graph coloring

algorithm can acheive order optimal capacity gains.

Remark 3.5.4. We have shown our proposed algorithm in pseudo-code in Algorithm 1. As we mentioned earlier, in our solution we only focus on content delivery. For cache placement, we assume that the greedy approximate algorithm proposed in [37] is used to populate helper caches. For helper assignment, the greedy algorithm is used. In other words, we suggest that the closest helper which has the content in its cache be assigned to the UT. During the content delivery phase, we use our proposed heuristic for index coding. We use a graph coloring heuristic for the conflict graph and find independent sets in the dependency graph. Then for each independent set only one multicast transmission is needed which will be sent by the helper. Our main contribution is to propose better content delivery algorithm compared to the baseline. In the decoding phase, UTs use their cached contents and the received content to decode their desired contents.

3.6 Simulations

In this section, we will show our simulation results. To show the performance of our coding technique, we have plotted the simulation results for five different sets of parameters in Figure 3.4. In this simulation, we assume that index coding is done in the packet level. We plotted the average packets sent in each transmission. We have assumed that the UTs are requesting contents based on a Poisson distribution with an average rate of $\frac{1}{n}$. This way we can assure that the average total request rate is one and we are efficiently using the time resource without generating unstable queues. The

Algorithm 1 Multihop Caching-Aided Coded Multicasting

```
procedure CACHE PLACEMENT
    Use greedy cache placement algorithm in [37]
    to populate helper caches.

procedure HELPER ASSIGNMENT
    Use a greedy algorithm to assign the closest helper
    which has the requested content, to the UT.

procedure CONTENT DELIVERY
    Form the dependency graph and conflict graph.
    Color the conflict graph by a graph coloring
    heuristic to find independent sets in dependency graph.
    for every independet set S found do
        Helper multicasts coded XOR of requested
        contents ⊕_{k∈S} M_{r_k}.

procedure CONTENT DECODING
    UT j XORs some of it's cached contents with the
    received coded content ⊕_{k∈S} M_{r_k} to decode M_{r_j}
```

optimum solution is an NP-hard problem. However, we used a very simple heuristic algorithm to count the number of cliques and cycles of maximum size 4. Even with this simple algorithm, we were able to show that the index coding can double the average number of packets per transmission in each time slot for certain values of the Zipfian parameter. Clearly, optimal index coding or more sophisticated algorithms can achieve better results compared to what we obtained by our simple algorithm.

For small values of s, the content request distribution is close to uniform, the dependency graph is very sparse and there is little benefit of using index coding. For large values of s, most UTs are requesting similar contents which results in broadcasting the same content to all nodes which is equal to one content per transmission. The main benefit of index coding happens for values of s between 0.5 and 2 which is usually the case in practical networks. Note that a wireless distributed caching system with no index coding, will always have one content per transmission.

Figure 3.5 compares the average number of requests satisfied per time slot by a helper between our proposed scheme and the baseline approach. For this simulation, a Zipfian content request distribution with parameter $s = 2$ is assumed. The cell radius is 400 meters, D2D transmission range is assumed to be 100 meters and 1000 UTs are considered in this figure similar to the simulations in [37] and Figure 3.1. Notice that with the baseline approach at least 27 helpers are required to cover the entire network and 24, 20, 14 and 10 helpers can cover 97%, 93%, 78% and 62% of the network, respectively. Using multihop D2D communications with a maximum of three hops, 4 helpers can cover the entire network while 3 helpers can cover 95% of the network

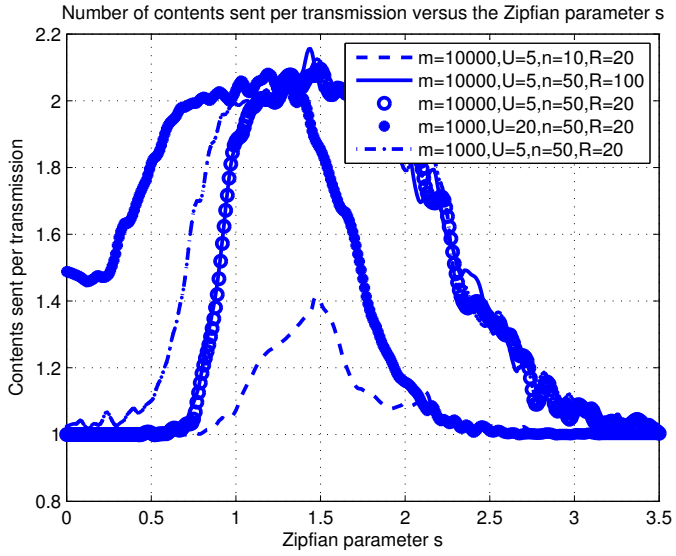

Figure 3.4: Average number of contents transmitted in each time slot when using index coding.

nodes. We have shown that even with our very simple heuristic algorithm, multihop D2D can significantly improve the helper utilization ratio. Note that in baseline approach, each helper can at most transmit one content per transmission, however, our approach can satisfy more than one request per transmission by taking advantage of the side information that is stored in nodes' caches. As the number of requests per user increases, there are more possibilities of creation of cliques of large size which results in increasing the efficiency of helper nodes. The simulation is carried for content request probability of up to 0.3 since realistically, no more than one third of nodes at any given time, request contents in the network.

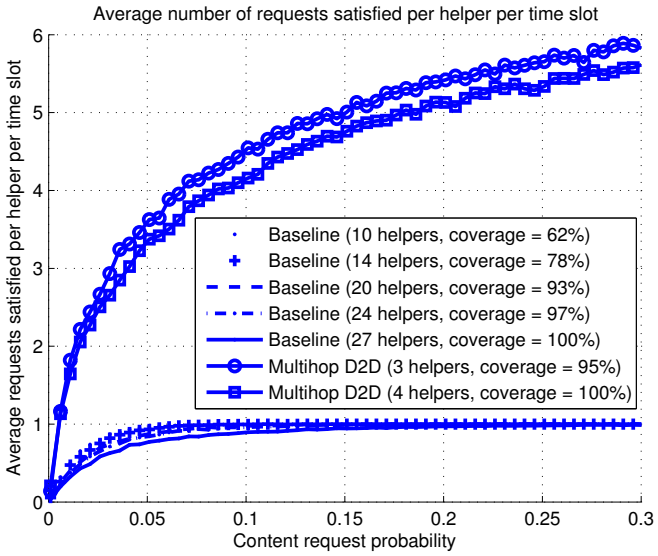

Figure 3.5: Comparing our proposed and baseline solution on average number of requests satisfied per time slot per helper versus the content request probability.

Chapter 4

Fountain Coding Based Caching in Cellular Networks

In this chapter, the idea of decentralized coded content caching for next generation cellular networks is studied. The contents are linearly combined and cached in under-utilized caches of User Terminals (UTs) and its throughput capacity is compared with decentralized uncoded content caching. In both scenarios, we consider multihop Device-to-Device (D2D) communications and the use of femtocaches in the network. It is shown that decentralized coded content caching can increase the network throughput capacity compared to decentralized uncoded caching by reducing the number of hops needed to deliver the desired content. Further, the throughput capacity for Zipfian content request distribution is computed and it is shown that the decentralized coded content cache placement can increase the throughput capacity of cellular networks by a factor of $(\log(n))^2$ where n is the number of nodes served by a femtocache. The results

in this chapter are published in [52] and [57].

4.1 Motivation

Recent advances in storage technology have made it possible for many consumer and user electronic products with Terabyte of storage capability. Many researchers are investigating the possibility of reusing this under-utilized storage capability to cache popular contents in order to improve the content delivery in cellular networks.

In recent years, the problem of caching has been extensively studied. The fundamental limits of caching in broadcast channels is studied in [72]. Other researchers [38, 49, 73, 80] extended the results in [72] for different scenarios in broadcast channels. The common features of all these studies are the assumptions that contents are cached without any coding and it is one hop communications. The authors in [44, 45] analyzed the capacity of multihop networks but they still assumed contents are cached without any coding, i.e., uncoded caching. Further, these studies [44, 45] focus on wireless ad hoc networks and there is no extension of the work to cellular networks.

In this chapter, we propose a radically different cache placement approach. While in our proposed algorithm each UT caches independently of all other UTs in a decentralized manner, redundant caching is avoided by storing a random bitwise XOR combination of popular contents. We call this method *decentralized coded content cache placement* algorithm. This approach increases the network throughput capacity and

does not suffer from over-caching problem of uncoded caching.

The proposed coded caching is fundamentally different from the notion of coded caching in references like [38, 49, 73, 80]. In such papers, during the cache placement phase, only uncoded contents or uncoded parts of contents are stored in the caches. Later during the content delivery phase, the base station broadcasts coded contents (linear combination of multiple contents) to UTs such that they can decode their files simultaneously. We instead propose that during the cache placement phase, the contents are randomly combined and cached in UTs. It is shown that this coded caching approach performs near optimal in terms of the average number of hops to retrieve a content and hence, it can significantly increase the network throughput capacity. This makes the proposed coded cache placement very suitable in practical systems where UTs have small storage capability compared to the total number of contents in the network.

Many studies propose to utilize high bandwidth D2D communications for UTs. Current IEEE 802.11ad standard [4] and the millimeter-wave proposal for future 5G networks [12, 84] are examples of such high bandwidth D2D communications. Authors in [53] extended the solution in [37] to deliver the contents from the helpers to the UTs using multihop D2D communications. However, [53] only considers uncoded caching.

We study our approach within the framework of future cellular networks that use femtocaches (or helpers) [37]. In such networks, several *helpers* with high storage capabilities are deployed in each cell to create a distributed wireless caching infrastructure. Each helper is serving a wireless ad hoc network of UTs through multihop D2D communications. We assume that helpers are connected to the base station through a

high bandwidth backhaul infrastructure. For simplicity of our analysis, we assume that the contents have equal sizes. Our results are valid for contents with different sizes since in practice each content can be divided into equal chunks. We prove that the proposed decentralized coded content caching increases the capacity of cellular networks by a factor of $(\log(n))^2$ compared to decentralized uncoded caching. As far as we know, work is the first work to propose the idea of decentralized coded content caching and to prove that coded cache placement can increase the network capacity.

The rest of this chapter is organized as follows. In section 4.2, the related work is discussed and section 4.3 describes the network model considered in this chapter. Section 4.4 focuses on the computation of the throughput capacity of wireless cellular networks operating under a decentralized uncoded cache placement algorithm and section 4.5 reports the capacity of coded cache placement algorithm. In section 4.6, we compute the capacity of networks operating under a Zipfian content request distribution. Simulation results are shown in section 4.7. Section 4.8 compares this work with other coding schemes.

4.2 Related Work

The original femtocache network model [37, 87] was focused on delivery of contents from femtocaches to UTs using single hop communications. The authors in [53] considered a femtocaching network with multihop D2D relaying of information from the helper to the UTs. A solution based on index coding was proposed in which the helper

utilizes the side information in the UTs to create index codes which are then multicasted to the UTs. The approach reduces bandwidth utilization by grouping multiple unicast transmissions into multicast transmission. While [53] proposed a solution for the helpers to efficiently multicast the contents to the UTs, this chapter assumes that the helper only unicasts the contents to the UTs. The UTs cache uncoded contents in [53] while in this chapter a decentralized coded content caching solution for UTs is proposed.

Caching has been a subject of interest to many researchers. The fundamental information theoretical limits of caching is studied in [72] where the authors studied the problem of caching in broadcast channels with a central uncoded cache placement algorithm. The authors in [73] extended the work of [72] to distributed uncoded cache placement approach and then broadcasting coded contents during the delivery phase over the shared link. They [73] proposed to break the contents into parts and then the UTs randomly cache the content parts during placement phase. In the delivery phase, coded contents are broadcasted from the server such that the UTs can decode their desired contents optimally. In this chapter we propose to randomly and independently combine contents and store them during cache placement phase. During the delivery phase, a linear combination of encoded files is used to retrieve the requested content. The notion of coded caching in [72] and all the papers that followed [38, 49, 79, 80] refers to broadcasting coded contents during the delivery phase and it is not a cache placement technique. All prior works [38, 49, 72, 73, 79, 80] are fundamentally different from this work as they are studying the information theoretic bounds of caching of a single server connected to users through a shared link while our work studies the scaling behavior of

networks in which the UTs retrieve requested contents through multiple hops.

Other works [44–47] studied the problem of caching in wireless and D2D networks. Authors in [46] discussed the fundamental capacity of D2D communication. In [47], a single hop D2D caching system is studied from an information theoretic point of view. The authors in [44, 45] have studied the capacity of multihop wireless D2D ad hoc networks with uncoded caching in certain regimes. However, our work is essentially different from [44, 45] in the sense that the UTs in our work always request the contents from the helper while in [44, 45], a wireless ad hoc network is considered. Clearly, such network model requires higher overhead to locate the route to the requested content while in our approach, the request always is sent toward the helper. They find the capacity for specific regimes where the cache size is relatively very large compared to the number of UTs. In this chapter, we prove that even constant cache size provides considerable capacity gain. Another major difference between our work and the references [44–47] is the introduction of decentralized coded content caching which has not been studied in these works.

Caching coded contents has been previously suggested [23, 65] as an efficient caching technique for devices with small storage capacity. In [65], the problem of index coding with coded side information is studied and [23] proposed a coded caching strategy for systems with small storage capacity. Our results demonstrate that apart from the practical importance of coded caching benefits for small storage devices, it can also increase the throughput capacity of cellular networks.

4.3 Network Model

In this chapter, we study the capacity of cellular networks utilizing a distributed femtocaching infrastructure as proposed in [37]. We assume several helpers with high storage capabilities are deployed throughout the network to assist in delivering the contents through multiple hops to UTs.

For capacity analysis, we use the deterministic routing approach proposed in [62]. Without loss of generality, it is assumed that the UTs are distributed on a square of area one and the helper is located at the center serving n UTs which are randomly distributed on a square. The square is divided into many square-lets of side length $c_1 s(n)$ where $s(n) \triangleq \sqrt{\frac{\log(n)}{n}}$. It is proved [82] that this network is connected if nodes have a transmission range of $\Theta\left(s(n)\right)$. When a UT requests a content from the helper, the content is routed [62] from the helper to the UT in a sequence of horizontal and vertical straight lines through square-lets which connect the helper to the UT.

A *Protocol Model* is considered [103] for successful communication between UTs. According to this model, if the UT i is located at Y_i, then a transmission from i to UT j is successful if $|Y_i - Y_j| < s(n)$ and for any other UT k transmitting on the same frequency band, $|Y_k - Y_j| > (1+\Delta)s(n)$ for a fixed guard zone factor Δ. A Time Division Multiple Access (TDMA) scheme is assumed for the transmission between the square-lets. With the assumption of Protocol Model, then the square-lets with a distance of $c_2 = \frac{2+\Delta}{c_1}$ square-lets apart can transmit simultaneously without significant interference [62].

The results are computed in terms of scaling laws. We use [59] the following order notations. Denote $f(n) = \mathrm{O}(g(n))$ if there exist $c > 0$ and $n_0 > 0$ such that $f(n) \leq cg(n)$ for all $n \geq n_0$, $f(n) = \Omega(g(n))$ if $g(n) = \mathrm{O}(f(n))$, and $f(n) = \Theta(g(n))$ if $f(n) = \mathrm{O}(g(n))$ and $g(n) = \mathrm{O}(f(n))$.

The contents in the network are represented by a set $\mathcal{X} = \{x_1, x_2, \ldots, x_m\}$ and the set of their indices by $\xi = \{1, 2, \ldots, m\}$. Without loss of generality, we assume that the contents with lower indices are more popular than the ones with higher indices. The contents are categorized into two groups of popular and less popular contents.

Definition 4.3.1. Define the set of h most popular contents as $\mathcal{X}_h = \{x_1, x_2, \ldots, x_h\} \subseteq \mathcal{X}$ where $\xi_h = \{1, 2, \ldots, h\} \subseteq \xi$ denotes the set of indices of the most popular contents.

The number of cached popular contents during the cache placement phase, h, is decided by the cellular network designer based on different parameters and specifications of the network. The selection of h is critical in the frequency of broadcast of unpopular contents by the base station. Typically h is chosen large enough such that with a very low probability contents are broadcasted from the base station. We assume that m and h grow polynomially with n similar to [44, 45]. Since h is a small fraction of m, we assume that h is growing with n in a much slower rate compared to m. Section 4.6 describes the necessary growth rate of h to guarantee that the probability of requesting a content with index larger than h decays polynomially with n with a decay rate of ρ. The results are general in nature because by allowing h and m scale with n with

different values of exponents, all possible values of h and m are considered.

The UTs have the same cache size of M and $M < h$[1]. There is no restriction on the cache size M and M is a constant or a function of n as in [44, 45]. During the cache placement phase, UT caches are filled independently of other UTs.

Helpers are assumed to store all the popular contents in $\mathcal{X}_h$. The popular content requests are served by D2D multihop communications and the less popular content requests are served by the base station through the low bandwidth shared link. The achievable throughput and network capacity are defined as follows.

Definition 4.3.2. A network throughput of $\lambda(n)$ contents per second for each UT is *achievable* if there is a scheme for scheduling transmissions in the D2D multihop network, such that every popular content request from $\mathcal{X}_h$ by every UT at a rate of $\lambda(n)$ can be served by the D2D multihop network.

Definition 4.3.3. The *throughput capacity* of the network is lower bounded by $\Omega(g(n))$ contents per second if a deterministic constant $c_3 > 0$ exists such that

$$\lim_{n\to\infty} \mathbb{P}[\lambda(n) = c_3 g(n) \text{ is achievable }] = 1. \tag{4.1}$$

The network throughput capacity is upper bounded by $O(g(n))$ contents per second if a deterministic constant $c_4 < +\infty$ exists such that

$$\liminf_{n\to\infty} \mathbb{P}[\lambda(n) = c_4 g(n) \text{ is achievable }] < 1. \tag{4.2}$$

The network throughput capacity is of order $\Theta(g(n))$ contents per second if it is lower bounded by $\Omega(g(n))$ and upper bounded by $O(g(n))$.

[1]Otherwise, the maximum throughput capacity is trivially achievable by caching all the popular contents in each UT.

We assumes that the cache placement is already done and we want to study the throughput capacity during the content delivery phase. If the content can be decoded using the cached information in the intermediate relaying UT caches, then the UT does not need to receive the content from the helper. However, if the content cannot be decoded using the intermediate relays, then the content is received from the helper.

To simplify the analysis, all contents are assumed of equal size with each having Q bits. This is a reasonable assumption since in practice the contents are divided into equal-sized chunks. The minimum number of hops required to successfully decode any content is denoted by Y with the average value of Y taken over all possible content requests denoted by $\mathbb{E}[Y]$. If the maximum achievable network throughput is $\lambda(n)$, then the network can deliver $n\lambda(n)$ contents per second or equivalently, UTs can transmit $n\lambda(n)\mathbb{E}[Y]Q$ bits per second. There are exactly $\frac{1}{(c_2c_1s(n))^2}$ square-lets at any time slot available for transmission. The maximum number of bits that the network can deliver is equal to $\frac{W}{(c_2c_1s(n))^2}$ where W is the total available bandwidth. Therefore,

$$\lambda(n) = \frac{W}{n\mathbb{E}[Y]Q(c_2c_1s(n))^2} = \Theta\left(\frac{1}{\mathbb{E}[Y]\log n}\right). \tag{4.3}$$

Hence, to compute the maximum achievable network throughput, it is enough [50] to find the average number of transmission hops needed to deliver the popular contents. Let's denote the requested content index by r, the probability of requesting i^{th}content by $f(i) = \mathbb{P}[r = i]$ and the cumulative probability function by $F(i) = \mathbb{P}[r \leq i]$. This implies that

$$\mathbb{P}[r \in \xi_h] = \mathbb{P}[r \leq h] = F(h). \tag{4.4}$$

With uncoded caching, if a UT $\mathcal{U}$ requests a content, the content is delivered to $\mathcal{U}$ either by the helper or by a relay that caches the content and is located on the routing path between $\mathcal{U}$ and the helper. With coded caching, if a group of coded contents can be used to decode the content are available along the routing path between $\mathcal{U}$ and the helper, then the helper informs the UTs the decoding instructions. If sufficient coded files do not exist in the caches of the UTs between $\mathcal{U}$ and helper to decode the desired content, then the content is routed to $\mathcal{U}$ from the helper. Since the helper only stores the popular contents in ξ_h and the less popular contents are downloaded from the base station, then the average number of traveled D2D hops in the network can be written as

$$\mathbb{E}[Y] = \mathbb{E}[Y|r \in \xi_h]\mathbb{P}[r \in \xi_h] = \mathbb{E}[Y|r \in \xi_h]F(h). \tag{4.5}$$

For many web applications [14, 15], the content request popularity follows Zipfian-like distributions. Although we express our results in general form, we will later compute explicit capacity results assuming a Zipfian content popularity distribution. Our main result in proving the gain of coded caching over uncoded caching is independent of the content popularity distribution.

For Zipfian popularity distribution with parameter s, the probability of requesting a content with popularity index i is

$$f(i) = \mathbb{P}[r = i] = \frac{i^{-s}}{\sum_{j=1}^{m} j^{-s}} = \frac{i^{-s}}{H_{m,s}}, \tag{4.6}$$

where $H_{m,s}$ represents the generalized harmonic number with parameter s. The rest of this chapter is dedicated to computing the throughput capacity of both decentral-

ized uncoded and coded content caching including for special case of Zipfian content popularity distribution.

4.4 Decentralized Uncoded Content Caching

This section focuses on throughput analysis in cellular networks when each UT cache M popular contents drawn uniformly at random independently of all other UTs. The uniform distribution of cache placement is different from the content request distribution by UTs. We will study the network throughput assuming a fixed cache placement.

Lemma 4.4.1. If a content is requested independently and uniformly at random from h most popular contents in $\mathcal{X}_h$, then the average required number of requests to have at least one copy of each content is equal to

$$\mathbb{E}_{\text{coupon collector}} = h \sum_{i=1}^{h} \frac{1}{i} = hH_h = \Theta(h \log(h)), \tag{4.7}$$

where $H_h = \Theta(\log(h))$ is the hth harmonic number.

Proof. This is known as the coupon collector problem [34]. □

Lemma 4.4.2. If each UT caches M different contents uniformly at random during cache placement phase, then the average number of UTs required to have at least one copy of each content in the network is

$$\frac{hH_h}{d(h, M)} \leq \mathbb{E}_{\text{uncoded}} \leq 1 + \frac{hH_h}{d(h, M)}, \tag{4.8}$$

where

$$d(h, M) \triangleq \sum_{j=0}^{M-1} \frac{h}{h-j}. \tag{4.9}$$

Proof. This is the extension of the coupon collector problem because the cached contents in each UT are different. To compute the average number of UTs in this case, we start from a classic coupon collector problem. Assume that contents are chosen uniformly at random and as soon as M different contents are found, they are cached in the first UT. Then the same process starts over for the next UT and after finding M different contents, the contents are placed in the UT's cache. Assume that this process is repeated until one copy of each content is cached in at least one UT's cache. Since this is a geometric distribution, on average we need $d(h, M)$ content requests to fill up one UTs cache with M different contents. Based on Lemma 4.4.1, after an average of hH_h content requests, we have requested one copy of all contents. Hence, the average number of UTs required to have one copy of each content in at least one UT cache is between $\frac{hH_h}{d(h, M)}$ and $1 + \frac{hH_h}{d(h, M)}$. □

Theorem 4.4.3. If $h \log(h) = O(Ms(n)^{-1})$[2], then the average number of transmission hops to receive the contents in uncoded caching is equal to

$$\mathbb{E}[Y|r \in \xi_h] = \Theta\left(\frac{h \log(h)}{M}\right). \tag{4.10}$$

[2]This condition means that the average number of hops needed to find the content is less than the number of hops to the helper. If this condition does not hold, then the content is sent by the helper to the requesting UT.

Proof. Lemma 4.4.2 shows that the average number of UTs needed so that all of the requests can be satisfied is

$$\frac{hH_h}{d(h,M)} \leq \mathbb{E}[Y|r \in \xi_h] \leq 1 + \frac{hH_h}{d(h,M)}. \tag{4.11}$$

Therefore, for large values of h, the average number of UTs required for finding any content scales as

$$\mathbb{E}[Y|r \in \xi_h] = \Theta\left(\frac{hH_h}{d(h,M)}\right). \tag{4.12}$$

To find a bound for $d(h,M)$, notice that the series in the right hand side of equation (4.9) has M terms and the maximum and minimum values are $\frac{h}{h-M+1}$ and 1 respectively. Therefore, $d(h,M)$ lower and upper bounds are

$$M \leq d(h,M) \leq \frac{Mh}{h-M+1}. \tag{4.13}$$

Using equations (4.12) and (4.13) we can find upper and lower bounds on $\mathbb{E}[Y|r \in \xi_h]$ as

$$\mathbb{E}[Y|r \in \xi_h] = O\left(\frac{h\log(h)}{M}\right), \tag{4.14}$$

$$\mathbb{E}[Y|r \in \xi_h] = \Omega\left(\frac{h\log(h)(h-M+1)}{Mh}\right). \tag{4.15}$$

Assuming $h >> M$, we have $h - M + 1 \approx h$ and the lower bound in (4.15) becomes equal to the upper bound (4.14) which proves the theorem. □

Figure 4.1 shows that, each request can be satisfied by another UT on average $\Theta(h\log(h)/M)$ hops away provided that $h\log(h) = O(Ms(n)^{-1})$. Theorem 4.4.3 and Equations (4.3) and (4.5) can be used to prove the following corollary.

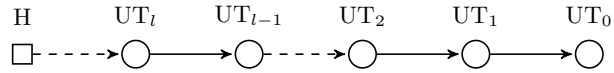

Figure 4.1: UT_0 is requesting a content which is available in another UT l hops away along the path toward helper H.

Corollary 4.4.4. The throughput capacity of the decentralized uncoded content caching network with $h\log(h) = O(Ms(n)^{-1})$ is upper bounded by

$$\lambda_{\text{uncoded}}(n) = O\left(\frac{M}{h\log(h)F(h)\log n}\right). \tag{4.16}$$

The throughput capacity in equation (4.16) is an upper bound and cannot be achieved because of congestion. The achievable throughput capacity will be computed in the following section.

4.5 Decentralized Coded Content Caching

The throughput capacity of decentralized coded content caching is computed in this section. We propose a random coding strategy and prove that the throughput capacity of the network is increased compared to uncoded caching strategy.

4.5.1 Coded cache placement

We assume the files are binary and all operations are in GF(2). For each encoded file, the helper randomly selects each one of the contents from the set $\mathcal{X}_h$ with probability $\frac{1}{2}$ and then combines all the selected contents (XOR) to create one encoded file. The encoded cached content at the jth cache location of UT i can be represented

as

$$f_j^i = \sum_{k=1}^{h} a_k^{ij} x_k = \mathbf{v}_j^i \mathbf{X}, \tag{4.17}$$

where $\mathbf{X} = [x_1 \ x_2 \ \dots \ x_h]^T$ is a column vector containing all popular contents of set $\mathcal{X}_h$ and $\mathbf{v}_j^i$ is a uniformly distributed encoding vector with binary elements and the summation is carried over GF(2). For a UT with cache size M, the helper creates M such encoded files. Therefore, each one of the contents in $\mathcal{X}_h$ has been used on average $\frac{M}{2}$ times to create the M coded files. We will represent the coded contents in caches as vectors belonging to $\mathbb{F}_2^h$.

4.5.2 Coded file reconstruction

The UT sends the request for a content to the helper. The helper then decides to send the file through a routing path as proposed in [62]. However, it is highly possible that the content can be reconstructed using a linear combination of some coded files in the caches of UTs between the requesting UT and the helper along the routing path. If these encoded files contain h linearly independent encoded vectors, they can span the entire message space. As depicted in Figure 4.2, UT_i in the routing path can contribute up to M linearly independent vectors $\mathbf{v}_1^i, \mathbf{v}_2^i, \dots, \mathbf{v}_M^i$ for decoding of a content. Therefore, UT_i which is at most q hops away from UT_0 on the routing path applies gain $b_j^i \in \mathrm{GF}(2)$ to its j-th cache content and then passes it down to the next hop closer to UT_0. This process of relaying and clever use of the caching contents continues hop by hop until the file reaches the requesting UT. After the requesting UT receives $(\sum_{i=1}^{q} \sum_{j=1}^{M} b_j^i \mathbf{v}_j^i)\mathbf{X}$, it can reconstruct its desired content by applying its own coding

gains to construct $(\sum_{i=0}^{q}\sum_{j=1}^{M} b_j^i \mathbf{v}_j^i) = e_k$ where e_k is a vector with all elements equal to zero except the kth element, i.e., x_k is reconstructed.

In this distributed decoding scheme, each relay UT adds some encoded files to the received file and relay it to the next hop. The coefficients $b_j^i \in \{0, 1\}$ are selected such that the linear combinations of encoded files produce the desired requested content. Each relay UT that participates in this distributed decoding operation, receives M bits from helper in order to combine its cached encoded files. The computational complexity for each UT is modest since it only involves XOR operation. The following lemma computes the average number of vectors $\mathbf{v}_j^i$ to create h linearly independent vectors.

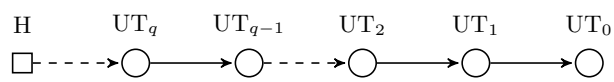

Figure 4.2: Each requested content by UT_0 is constructed by a linear combinations of the contents in $q+1$ UTs caches on the path between the helper and UT_0.

Lemma 4.5.1. Let $\mathbf{v}_j^i$ be a random vector belonging to $\mathbb{F}_2^h$ with binary elements with uniform distribution. The average number of vectors $\mathbf{v}_j^i$ to span the whole space of $\mathbb{F}_2^h$ equals

$$\mathbb{E}_v = h + \sum_{i=1}^{h} \frac{1}{2^i - 1} = h + \gamma, \tag{4.18}$$

where γ asymptotically approaches the Erdős–Borwein constant (≈ 1.6067)[3].

Proof. We use a Markov chain to model the problem. The states of this Markov chain are equal to the dimension of the space spanned by vectors[4] $v_1, v_2, \ldots, v_l$. Let k_l ($k_l \leq h$)

[3]In our problem, h is large enough that we can approximate the summation in (4.18) with asymptotic value.

[4]For the rest of lemma, we drop superscript from notations where it is obvious.

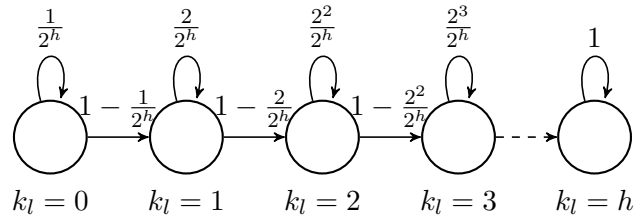

Figure 4.3: The state space of the Markov chain used in proof of Lemma 4.5.1.

denote the dimension of the space spanned by these vectors. Therefore, the Markov chain will have $k_l + 1$ distinct states. Assuming that we are in state k_l, we want to find the probability that adding a new vector will change the state to $k_l + 1$. When we are in state k_l, 2^{k_l} vectors out of 2^h possible vectors will not change the dimension while adding any one of $2^h - 2^{k_l}$ new vectors will change the dimension to $k_l + 1$. Therefore, the Markov chain can be represented by the one in Figure 4.3. The state transition matrix for this Markov chain is

$$P = \begin{bmatrix} \frac{1}{2^h} & 1-\frac{1}{2^h} & 0 & \cdots & 0 & 0 \\ 0 & \frac{2}{2^h} & 1-\frac{2}{2^h} & \cdots & 0 & 0 \\ 0 & 0 & \frac{2^2}{2^h} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1-\frac{2^{h-2}}{2^h} & 0 \\ 0 & 0 & 0 & \cdots & \frac{2^{h-1}}{2^h} & 1-\frac{2^{h-1}}{2^h} \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix},$$

which can be written in the form of a discrete phase-type distribution as

$$P = \begin{bmatrix} T & T_0 \\ \underline{0} & 1 \end{bmatrix}, \tag{4.19}$$

where

$$T = \begin{bmatrix} \frac{1}{2^h} & 1-\frac{1}{2^h} & 0 & \cdots & 0 & 0 \\ 0 & \frac{2}{2^h} & 1-\frac{2}{2^h} & \cdots & 0 & 0 \\ 0 & 0 & \frac{4}{2^h} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \frac{2^{h-2}}{2^h} & 1-\frac{2^{h-2}}{2^h} \\ 0 & 0 & 0 & \cdots & 0 & \frac{2^{h-1}}{2^h} \end{bmatrix}, \tag{4.20}$$

and

$$T_0 = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1-\frac{2^{h-1}}{2^h} \end{bmatrix}. \tag{4.21}$$

If e denotes all one vector of size h, since P is a probability distribution we have

$$P \begin{bmatrix} e \\ 1 \end{bmatrix} = \begin{bmatrix} e \\ 1 \end{bmatrix}. \tag{4.22}$$

This implies that $Te + T_0 = e$, hence $T_0 = (I - T)e$. Therefore, it is easy to show by

induction that the state transition matrix in l steps can be written as

$$P^l = \begin{bmatrix} T^l & (I - T^l)e \\ 0 & 1 \end{bmatrix}. \tag{4.23}$$

This equation implies that if we define the absorption time[5] as

$$t_a = \inf\{l \geq 1 \mid k_l = h\}, \tag{4.24}$$

and if l is strictly less than the absorption time, the probability of transitioning from state i to state j by having l new vectors can be computed from T^l. In other words,

$$\mathrm{P}_i^l[k_l = j, l < t_a] = (T^l)_{ij}. \tag{4.25}$$

Therefore, starting from state i, if t_j^i denotes the number of vectors (i.e., encoded files for our problem) to transition from state j before absorption, t_j^i can be written as

$$t_j^i = \sum_{l=0}^{t_a-1} 1\{k_l = j\}. \tag{4.26}$$

The average value of t_j^i is

$$\mathbb{E}[t_j^i] = \mathbb{E}\left[\sum_{l=0}^{t_a-1} 1\{k_l = j\}\right] = \sum_{l=0}^{t_a-1} \mathbb{E}\left[1\{k_l = j\}\right]. \tag{4.27}$$

Since $\mathbb{E}\left[1\{k_l = j\}\right] = \mathrm{P}_i^l(k_l = j, l \leq t_a - 1)$, we have

$$\begin{aligned} \mathbb{E}[t_j^i] &= \sum_{l=0}^{t_a-1} \mathrm{P}_i^l(k_l = j, l \leq t_a - 1) \\ &\overset{a}{=} \sum_{l=0}^{\infty} \mathrm{P}_i^l(k_l = j, l < t_a) \end{aligned}$$

[5]In our problem, absorption time is actually the total number of required vectors in relays to span the h-dimensional space.

$$\stackrel{b}{=} \sum_{l=0}^{\infty} (T^l)_{ij}. \tag{4.28}$$

Equality (a) is correct because the probability is nonzero up to $l = t_a - 1$ terms and (b) is derived from equation (4.25). If we denote matrix $U = (\mathbb{E}[t_j^i])_{ij}$, by using equation (4.27) and matrix algebra, we have

$$U = \sum_{i=0}^{\infty} T^i = (I - T)^{-1}. \tag{4.29}$$

It is not difficult to verify that

$$U = (I-T)^{-1} = \begin{bmatrix} \frac{2^h}{2^h-1} & \frac{2^{h-1}}{2^{h-1}-1} & \frac{2^{h-2}}{2^{h-2}-1} & \cdots & 2 \\ 0 & \frac{2^{h-1}}{2^{h-1}-1} & \frac{2^{h-2}}{2^{h-2}-1} & \cdots & 2 \\ 0 & 0 & \frac{2^{h-2}}{2^{h-2}-1} & \cdots & 2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 2 \end{bmatrix}.$$

We are interested in computing the average number of vectors (t_a) to get to absorption starting from $k_l = 0$. Hence,

$$\begin{aligned} \mathbb{E}_v = \mathbb{E}[t_a] &= \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix} Ue \\ &= \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix} (I-T)^{-1} e \\ &= \sum_{i=1}^{h} \frac{2^i}{2^i - 1} = h + \sum_{i=1}^{h} \frac{1}{2^i - 1} \end{aligned} \tag{4.30}$$

This proves the lemma. □

Remark 4.5.2. The optimal number of linearly independent vectors to span the vector space is h. Our decentralized coded content caching strategy only requires $h + \gamma$ coded

contents to span the vector space. Since γ is considerably smaller than h, then our approach provides close to optimal performance in terms of the minimum required number of caches.

Lemma 4.5.1 suggests that each UT's request can be satisfied in a smaller number of hops compared to an uncoded caching strategy. This shows that our proposed decentralized coded content caching scheme is capable of removing the inherent over-caching problem in decentralized uncoded content caching. The following theorem formalizes the result.

Theorem 4.5.3. If the number of popular contents is upper bounded by $Ms(n)^{-1}$ i.e. if $h = O(Ms(n)^{-1})$, then our proposed decentralized coded content caching technique reduces the average number of hops for decoding a content to

$$\mathbb{E}[Y|r \in \xi_h] = \Theta\left(\frac{h}{M}\right). \tag{4.31}$$

Proof. Lemma 4.5.1 shows that in order to decode a requested content, on average $\Theta(h)$ coded contents are required. Since each UT has a cache of size M, Lemma 4.5.1 shows that on average we need $\Theta(\frac{h}{M})$ UTs (hops) to be able to decode the desired content. Notice that each individual UT does not need to separately send their coded content to the requesting UT. Each UT combines its encoded files with a file that it receives from previous hop and forwards it to the next hop. □

Using the results of theorem 4.5.3 and equations (4.3) and (4.5), the throughput capacity of coded caching can be upper bounded as follows.

Corollary 4.5.4. If $h = O(Ms(n)^{-1})$, the throughput capacity of a decentralized coded content caching network is upper bounded by

$$\lambda_{\text{coded}}(n) = O\left(\frac{M}{hF(h)\log n}\right). \tag{4.32}$$

The throughput in the right hand side of equation (4.32) may not be achievable due to network congestion. In the following we will find achievable network throughputs.

Theorem 4.5.5. The decentralized uncoded and coded content caching strategies have the order throughput capacity of

$$\lambda_{\text{uncoded}}(n) = \Theta\left(\frac{1}{F(h)\log n}\left(\frac{M}{h\log(h)}\right)^2\right), \tag{4.33}$$

$$\lambda_{\text{coded}}(n) = \Theta\left(\frac{1}{F(h)\log n}\left(\frac{M}{h}\right)^2\right), \tag{4.34}$$

respectively when $h\log(h) = O(Ms(n)^{-1})$.

Proof. Clearly, the square-lets that are closer to the helper are more prone to traffic congestion. Hence, if the achievable throughput capacity of the square-let that contains the helper is computed, this capacity is also achievable in other square-lets. This square-let must respond to all requests which are located on average within a radius of $\Theta(h/M)$ hops away from it (or $\Theta(h\log(h)/M)$ hops in uncoded caching case). Therefore, it should be able to serve on average $\Theta(\log n(h/M)^2)$ (or $\Theta(\log n(h\log(h)/M)^2)$) requests. The probability that the popular contents are requested by UTs is $F(h)$. Thus, the average number of popular content requests from this square-let is $\Theta(\log n(h/M)^2F(h))$ (or $\Theta(\log n(h\log(h)/M)^2F(h))$).

Therefore, for the case of coded caching, this square-let should be able to deliver $\Theta(\lambda_{\text{coded}}(n)\ (h/M)^2 F(h)\log n)$ contents per second and $\Theta(\lambda_{\text{uncoded}}(n)(h\log(h)/M)^2 F(h)\log n)$ contents per second for uncoded caching. Since the network has a bandwidth of W and we use TDMA scheme, each square-let can only deliver $\Theta(1)$ contents per second. Therefore, both $\lambda_{\text{coded}}(n)(h/M)^2 F(h)\log n$ and $\lambda_{\text{uncoded}}(n)\ (h\log(h)/M)^2 F(h)\log n$ should scale as $\Theta(1)$. This proves the theorem. □

Since the throughput capacity in (4.33) (or (4.34)) is achievable throughput and less than equation (4.16) (or (4.32)), then equation (4.16) (or (4.32)) cannot be achieved and is only an upper bound.

Remark 4.5.6. Theorem 4.5.5 shows that coded caching increases the throughput capacity of the multihop femtocaching D2D network by a factor of $\Theta((\log(h))^2)$. Since h can be a function of n (as will be shown in the next section), it can be concluded that coded caching can increase the throughput capacity of the network up to a factor of $\Theta((\log n)^2)$.

Remark 4.5.7. In general, majority of the contents can be retrieved from other UTs. However, if a content cannot be retrieved from UTs on the path between the helper and the requesting UT, then the helper directly sends this content through multihop communication to the requesting UT. This can happen both in coded and uncoded caching schemes and it happens when the requesting UT is very close to the helper node.

4.6 Capacity of networks with Zipfian content request distribution

In this section, we compute the throughput capacity of Zipfian distribution by utilizing the results in sections 4.4 and 4.5. To proceed, we first prove the following lemma.

Lemma 4.6.1. Let μ and η be constants such that $\mu > \eta > 0$ and let $a(n)$ and $b(n)$ be two functions that scale as $\Theta(n^{\eta})$ and $\Theta(n^{\mu})$ respectively. Then, for $s > 1$ we have

$$\sum_{i=a(n)+1}^{b(n)} i^{-s} = \Theta(n^{-\eta(s-1)}). \tag{4.35}$$

Proof. Let $d(n) \triangleq \lfloor \frac{b(n)}{a(n)} \rfloor$. We have,

$$\begin{aligned}\sum_{i=a(n)+1}^{b(n)} i^{-s} &= \sum_{j=1}^{d(n)-1} \sum_{i=ja(n)+1}^{(j+1)a(n)} i^{-s} + \sum_{i=d(n)a(n)+1}^{b(n)} i^{-s} \\ &\leq \sum_{j=1}^{d(n)-1} \sum_{i=ja(n)+1}^{(j+1)a(n)} i^{-s}. \end{aligned} \tag{4.36}$$

Since for $ja(n) + 1 \leq i \leq (j + 1)a(n)$ we have $ja(n) < i$, then

$$\sum_{i=ja(n)+1}^{(j+1)a(n)} i^{-s} < \sum_{i=ja(n)+1}^{(j+1)a(n)} (ja(n))^{-s} = a(n)(ja(n))^{-s}. \tag{4.37}$$

Combining (4.36) and (4.37), we arrive at

$$\sum_{i=a(n)+1}^{b(n)} i^{-s} < a(n)^{-s+1} \sum_{j=1}^{d(n)-1} j^{-s} < a(n)^{-s+1}\zeta(s), \tag{4.38}$$

where $\zeta(s) = \sum_{i=1}^{\infty} i^{-s}$ denotes the Reimann zeta function and it is a constant value for $s > 1$. Therefore, the upper bound is given by

$$\sum_{i=a(n)+1}^{b(n)} i^{-s} = O(a(n)^{-s+1}) = O(n^{-\eta(s-1)}). \tag{4.39}$$

For the lower bound of this summation, we will use an integral approximation to derive the results.

$$\begin{aligned}\sum_{i=a(n)+1}^{b(n)} i^{-s} &\geq \int_{a(n)+1}^{b(n)} x^{-s}\mathrm{d}x \\ &= \frac{(a(n)+1)^{-s+1} - b(n)^{-s+1}}{s-1}\end{aligned} \tag{4.40}$$

Since, $a(n)$ scales as $\Theta(n^{\eta})$ and $b(n)$ scales as $\Theta(n^{\mu})$ and $\mu > \eta$, then the first term in the right hand side of equation (4.40) is dominant and we have

$$\sum_{i=a(n)+1}^{b(n)} i^{-s} = \Omega(n^{-\eta(s-1)}). \tag{4.41}$$

This proves the lemma. □

As mentioned in section 4.3, we assume that m is growing polynomially with n. Lets assume that h which is a tiny fraction of m also grows polynomially with n. We will now find the necessary growth rate of h with n to guarantee that the request probability for non-popular contents decays polynomially with n with a decay rate of $\rho > 0$. In other words, we want to find the necessary growth rate for h_ρ such that for constants c_6 and n_1 and for any $n > n_1$ we have

$$\mathbb{P}[r > h_\rho] \leq c_6 n^{-\rho}. \tag{4.42}$$

Assuming that the necessary growth rate for h_ρ is τ, i.e.$h_\rho = \Theta(n^\tau)$, using Lemma 4.6.1, we have

$$\sum_{i=h_\rho+1}^{m} i^{-s} = \Theta(n^{-\tau(s-1)}). \tag{4.43}$$

This means that there exist constant c_7 and n_2 such that for any $n > n_2$ we have

$$\sum_{i=h_\rho+1}^{m} i^{-s} \leq c_7 n^{-\tau(s-1)}. \tag{4.44}$$

Since $H_{m,s} \geq 1$, we arrive at

$$\mathbb{P}[r > h_\rho] = 1 - F(h_\rho) = \sum_{i=h_\rho+1}^{m} \frac{i^{-s}}{H_{m,s}} \leq c_7 n^{-\tau(s-1)}. \tag{4.45}$$

Therefore, to ensure that equation (4.42) remains valid, it is enough to choose c_7 equal to c_6 and $\tau = \frac{\rho}{s-1}$. Hence, to guarantee that (4.42) holds, h_ρ should scale as

$$h_\rho = \Theta(n^{\frac{\rho}{s-1}}). \tag{4.46}$$

Therefore, in a network where m grows polynomially with n with a rate of $\alpha > \frac{\rho}{s-1}$, if h_ρ grows as (4.46), the request probability for non-popular contents decays faster than $\Theta(n^{-\rho})$. For the rest of this section, we assume that ρ is a design parameter and the helper caches contents from among the $h_\rho = \Theta(n^{\frac{\rho}{s-1}})$ most popular contents.

Remark 4.6.2. If we choose h_ρ based on equation (4.46) such that it satisfies $h_\rho \log(h_\rho) = O(Ms(n)^{-1})$, equations (4.33) and (4.34) can be rewritten as

$$\lambda^\rho_{\text{uncoded}}(n) = \Theta\left(\frac{1}{\log n}\left(\frac{M}{h_\rho \log(h_\rho)}\right)^2\right), \tag{4.47}$$

$$\lambda^\rho_{\text{coded}}(n) = \Theta\left(\frac{1}{\log n}\left(\frac{M}{h_\rho}\right)^2\right). \tag{4.48}$$

Equations (4.47) and (4.48) show that coded caching can increase the throughput capacity of Zipfian networks by a factor of $(\log(h_\rho))^2$ which from equation (4.46), it implies a factor of $(\log n)^2$ increase in throughput capacity.

Theorem 4.6.3. For a network with non-heavy tailed Zipfian content request distribution ($s > 1$) such that the probability of content request from the base station decays polynomially with n with a rate of ρ and $h_\rho \log(h_\rho) = O(Ms(n)^{-1})$, then the following throughputs are achievable for the D2D network.

$$\lambda_{\text{uncoded}}^{\text{Zipf},\rho}(n) = \Theta\left(\frac{n^{-\frac{2\rho}{s-1}}}{(\log n)^3} M^2\right) \tag{4.49}$$

$$\lambda_{\text{coded}}^{\text{Zipf},\rho}(n) = \Theta\left(\frac{n^{-\frac{2\rho}{s-1}}}{\log n} M^2\right) \tag{4.50}$$

Proof. As mentioned earlier, in a Zipfian content request distribution with $s > 1$ to ensure that the probability of requesting non-popular contents decays polynomially with n with a rate of ρ, it is enough to choose h_ρ as in equation (4.46). If we use this value for h_ρ and plug it in equations (4.47) and (4.48), we will arrive at equations (4.49) and (4.50). □

4.7 Simulations

The simulation results compare the performance of proposed decentralized coded content caching with decentralized uncoded content caching. The helper serves $n = 1000$ UTs in the network with Zipfian content request probability with parameter

$s = 2$. In this network, $h = 100$ highly popular contents account for more than 99% of the total content requests. Our simulations are carried over a cell with radius 2000 meters and for a D2D transmission range of 10 meters. Figure 4.4 shows the simulation results comparing the average number of hops required to decode the contents in both decentralized coded and uncoded content caching schemes. The simulation results clearly demonstrate that decentralized coded cache placement outperforms uncoded case particularly when the cache size is small which is the usual operating regime. For instance, with decentralized coded content caching, a cache of size 20 only requires less than 5 hops while decentralized uncoded content caching needs around 22 hops for successful content retreival. This makes coded content caching much more practical compared to uncoded content caching. Note that capacity is inversely proportional to the average hop counts.

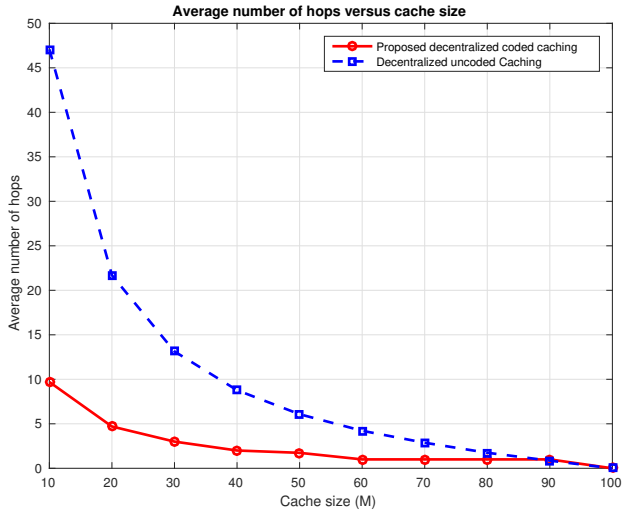

Figure 4.4: Simulation results for a helper serving 1000 UTs in a cell of radius 2000 meters with a D2D transmission range of 10 meters and a total of 100 popular contents.

As can be seen from Figure 4.4, for small cache sizes, coded cache placement significantly reduces the number of hops required to decode the contents. This property is important for systems with small storage capability for UTs since large number of hops can impose excessive delay and low quality of service.

Figure 4.5 compares throughput capacity of coded content caching with uncoded content caching. The content popularity request distribution is Zipfian with parameter $s = 2.5$. The results demonstrate significant capacity gain for decentralized coded content cache placement. The parameter $\rho = 0.75$ suggests that h scales as $\Theta(\sqrt{n})$ in this plot (equation (4.46)). Notice that in this plot, m can scale as $\Theta(n^{\alpha})$ where $\alpha > 0.5$ can potentially be a large number. As can be seen from Figure 4.5, for only 100 UTs and a constant cache size, throughput capacity of coded caching is 20 times higher than the throughput capacity of uncoded caching.

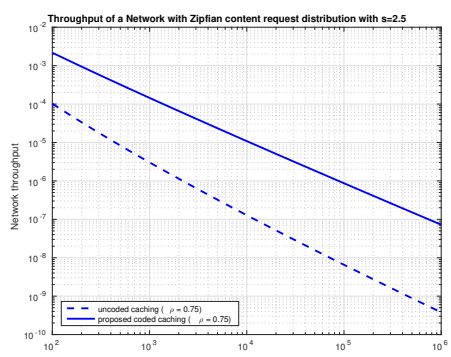

Figure 4.5: Network throughput capacity comparison of the decentralized coded content caching and decentralized uncoded content caching schemes.

4.8 Discussion

Decentralized coded caching uses uniform random vectors in $\mathbb{F}_2^h$. This approach can be considered as special case of LT-codes [90]. In fact, coded caching is a random LT-code with parameters $(h, \Omega(x))$ where $\Omega(x) = \frac{1}{2^h}(1+x)^h$ is the generator polynomial [69]. LT-codes are a practical realization of fountain codes which are proven to be very useful in erasure channels and storage systems. The uniform random LT-code in our work is using all the possible random vectors in $\mathbb{F}_2^h$ for encoding. This allows the receiver UT to decode any content by accessing $h + 1.6067$ cache locations (on average) which is very close to the optimal value of h. Using other types of LT-codes, the decoding cost can be reduced but they increase the number of required cache locations to decode the contents and therefore decrease the capacity compared to uniform random LT-codes. On the other hand, Raptor codes [90] which are another class of fountain codes can be used to perform the decoding in constant time with slightly fewer number of required cache locations (greater than or equal to h). However, if h is large, then the achieved throughput capacity with raptor codes is very close to our proposed technique.

The proposed approach is similar to Random Linear Network Coding (RLNC) [42]. In RLNC, each content is divided into chunks and those chunks are randomly coded and distributed in the network. In order for the user to decode the content, it must receive enough innovative packets such that it can decode those chunks. In our approach, we linearly combine different contents and only are interested in one of the contents. Therefore, in our decoding approach, we don't transmit all encoded files to

the receiver. Instead, the encoding information is sent hop by hop from the helper to the UTs. Each UT needs to use that encoding information to uniquely combine its cached files with the received file from previous hop and transmit it to the next hop for more processing. One can use coded content caching to store files and then use network coding to transfer these coded contents in the network instead of simply transmitting the entire encoded files.

Chapter 5

Fountain Coding Based Caching in Wireless Ad Hoc Networks

In this chapter we will propose a new caching technique for wireless ad hoc networks. Network caching aims to temporarily store data in some nodes within the network to be able to get the contents in shorter time periods. However, caching networks did not always consider secure storage (due to the compromise between time performance and security). In this chapter, a novel decentralized coded caching approach is proposed. In this solution, nodes only transmit coded files to avoid eavesdropper wiretappings and to protect the user contents. In this technique random vectors are used to combine the contents using XOR operation. We modelled the proposed coded caching scheme by a Shannon cipher system to show that coded caching achieves asymptotic perfect secrecy. The proposed coded caching scheme significantly simplifies the routing protocol in cached networks while it reduces overcaching and achieves a higher through-

put capacity compared to uncoded caching in reactive routing. It is shown that with the proposed coded caching scheme, any content can be retrieved by selecting a random path while achieving asymptotic optimum solution. We also study the cache hit probability and show that the coded cache hit probability is significantly higher than uncoded caching. A secure caching update algorithm is also presented. The results in this chapter are published in [56] and [55].

5.1 Motivation

Significant advances in wireless and mobile technologies over the past decades have made it possible for mobile users to stream high quality videos and to download large-sized contents. Video streaming applications like Netflix, Hulu and Amazon have become increasingly popular among mobile users. Close to half of all video plays were on mobile devices like tablets and smartphones[1] during the fourth quarter of 2015.

In parallel, storage capacity of mobile devices has significantly increased without being fully utilized. Efficient use of this under-utilized storage is specially important for video streaming applications that account for a large portion of the overall internet traffic.

On the other hand, *proprietary content* as opposed to *user-generated content* is the most important asset of many content sharing companies including Netflix, Hulu and Amazon. These companies use extreme measures to protect their data and therefore they are hesitant to let the end users cache their contents locally. One possible solution

[1] http://go.ooyala.com/rs/447-EQK-225/images/Ooyala-Global-Video-Index-Q4-2015.pdf

is to encrypt each content before caching it locally. Encryption algorithms reduce the content sharing rate. Further, such algorithms are only *computationally secure* and an adversary is able to break them with time. For instance, Data Encryption Standard (DES) which was once the official Federal Information Processing Standard (FIPS) in US is not considered secure anymore. In this chapter, we propose an *information theoretically* secure solution for caching proprietary contents which cannot be decoded with time.

Physical layer security in wireless networks has been the subject of many recent research papers. With a focus on physical layer security in wireless ad hoc networks, we propose a novel decentralized coded caching approach in which random vectors are used to combine the contents using XOR operation. Nodes only transmit coded files and therefore an eavesdropper with a noiseless channel cannot decode the desired contents. Our technique in achieving physical layer security is quite different from techniques which exploit wireless channel dynamics to achieve physical layer security. We demonstrate that coded caching technique is similar to Shannon cipher [88] problem and it can achieve asymptotic perfect secrecy during file transmissions by using a secure low bandwidth channel to exchange the decoding gains. The main contributions of this chapter are as follows.

- Introducing decentralized coded caching for wireless ad hoc networks.
- Computing capacity of coded caching for proactive and reactive routing strategies[2]

[2]Notice that the notion of proactive (or reactive) routing that we study is the extreme case when complete (or no) network knowledge is available.

and comparing the results with uncoded caching.

- Proving asymptotic perfect secrecy for coded caching scheme.
- Computing the cache hit probability for coded caching and comparing it with uncoded caching.
- Introducing coded caching update algorithm.

The rest of the chapter is organized as follows. In section 5.2, the related works are described. In section 5.3, the network model, proposed encoding and decoding and earlier results on coded caching are described. Section 5.4 focuses on the network capacity, while the security aspect of coded caching is studied in section 5.5. Cache hit probability for both coded and uncoded caching approaches is studied in section 5.6. In section 5.7, an efficient caching update algorithm is proposed. Simulation results are studied in section 5.8. We have discussed some of the implementation details of our approach in section 5.9.

5.2 Related Work

Many researchers have investigated the problem of caching in recent years. The fundamental limits of caching over a shared link is studied in [72]. The authors in [72] suggested to store uncoded contents or uncoded segments of the contents in the caches during the cache placement phase. Later on, during the content delivery phase, they proposed to broadcast coded files to the users which resulted in significant coded

multicasting gain. Such a gain is achievable by taking advantage of content overlap at the various caches in the network, created by a central coordinating server. This work was later extended in [73] to scenarios with decentralized uncoded cache placement and there are many other related works that followed the same set of assumptions. While references [72, 73] study the problem of caching in broadcast channels, we study the problem of caching in wireless ad hoc networks using multihop communications.

The authors in [45] studied the problem of caching in multihop networks. They assumed that during the cache placement phase, uncoded contents are stored in the caches independently in a decentralized fashion. They studied the throughput capacity of wireless ad hoc networks. In this chapter, we introduce decentralized coded caching for wireless ad hoc networks and compare our results with uncoded caching results in [45].

In decentralized uncoded caching scheme that we have considered in this chapter, we assumed that different contents are selected uniformly at random and placed in the caches during the cache placement phase. We can therefore model the uncoded caching strategy as a coupon collector problem with group drawings as studied in [91] and [48]. Similar to the classical coupon collector problem, it is proved in [48] that uniform selection of cached contents results in the minimum possible value for the average number of required hops. Hence, in this chapter we have studied the case of uniform cache placement which is the best possible scenario in terms of network capacity.

In the proposed decentralized coded caching scheme in this chapter, we have used random uniform LT codes [69,71,90] to encode the contents during cache placement.

In LT coding-based cache placement, an LT code is chosen and then the contents are encoded using the selected LT code. Then such encoded files are stored in the nodes.

In random binary uniform LT codes [71] a random code of length m is chosen from $\mathbb{F}_2^m$ such that any of its elements is either equal to zero or one. Such a random code can be represented by a random vector of length m in $\mathbb{F}_2^m$. In our proposed coded caching approach, contents with indices corresponding to one in the random vector are added together and the encoded files are cached in the nodes.

Physical layer security has attracted many researchers in recent years. A survey of recent progress in this field can be found in [11,89]. Security aspects of network coding is studied in references like [16, 67, 95]. In this chapter, we will study the security of LT coding-based caching technique. We specifically investigate the last hop communication security which is the most vulnerable transmission link. We prove that for a certain regime of caching sizes in the network, asymptotic perfect secrecy is achievable for the last hop[3]. Further, a secure caching update algorithm is proposed.

Our proposed decentralized caching scheme can be used to create a distributed network coding based storage system. The concept of network coding for distributed storage was originally studied in [27, 29] where files are divided into packets and nodes need to collect all the packets to be able to reconstruct their desired contents. In our proposed distributed caching scheme, the requesting node needs to decode only one content. The authors in [27, 29] study the repair problem which is the problem of recovering data when some nodes fail. Further, while they address the repair problem

[3]Security investigation for other hops remains as future work.

using MDS codes, our proposed coded caching technique is based on LT codes [69]. This chapter focuses on the scaling capacity and security of a network with decentralized coded caching which is not studied in previous references [27, 29].

LT coding based storage is studied in references like [17, 99]. Reference [17] investigates the repair problem of LT codes in cloud services and [99] proposes new types of LT codes for storage systems. Use of fountain codes and Raptor codes [90] has also been studied in [28, 40, 61, 96, 97]. None of these references have studied capacity, cache hit probabilities, cache update algorithms and security of LT code based storage in wireless ad hoc networks.

5.3 Preliminaries

5.3.1 Network Model

We consider a dense wireless ad hoc network in which n nodes are uniformly distributed over a square of unit area as depicted in Figure 5.1. These nodes use multihop communications to request one of m contents from a set labeled as $\mathcal{F} = \{F_1, F_2, \ldots, F_m\}$. The requested content is denoted by F_r and the minimum number of nodes to decode F_r by N_r. The minimum number of nodes to decode any requested content is denoted by N. We also assume that each node can cache M files of equal size each containing Q bits. In practice, the contents are divided into equal-sized chunks and the chunks are cached. Such an assumption is common in many references including [53, 72, 73].

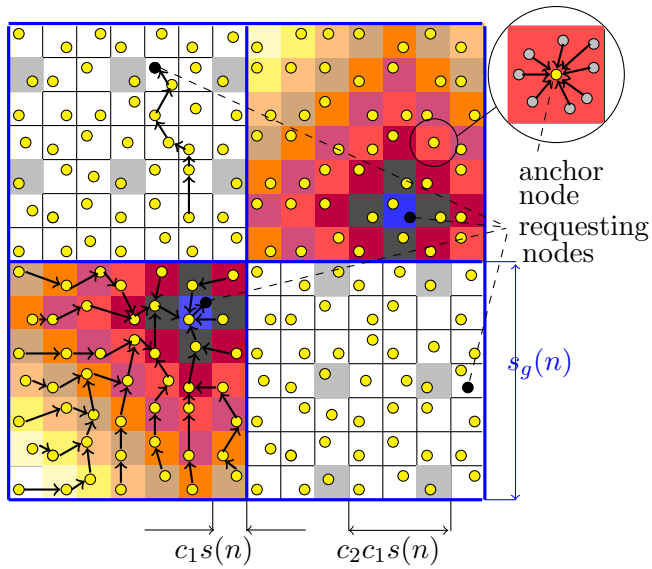

Figure 5.1: Each local group with side length $s_g(n)$ contains many square-lets of side length $c_1 s(n)$. Each square-let has one randomly selected anchor node.

The unit square area in Figure 5.1 is divided into many square-lets each with a side length of $c_1 s(n)$ where $s(n) = \sqrt{\log(n)/n}$. It is shown [62] that each square-let contains $\Theta(\log(n))$ nodes with probability 1. To avoid multiple access interference, a Protocol Model [103] is considered for successful communication between nodes. A Time Division Multiple Access (TDMA) scheme is assumed for the transmission between the square-lets. With the assumption of Protocol Model, if each square-let has a side length of $c_1 s(n)$ for a constant c_1, then the square-lets with a distance of $c_2 = \frac{2+\Delta}{c_1}$ square-lets apart can transmit simultaneously without significant interference [62] for a constant value Δ.

For our proactive routing analysis, we assume that one of the nodes in each square-let is randomly chosen which is called anchor node. This node collects all the information in the square-let and combines them and relays the coded information to

the next hop toward the requesting node. It is known that a minimum transmission range of $\Theta\left(\sqrt{\frac{\log(n)}{n}}\right)$ ensures network connectivity [50] in such a dense network.

For the case of uncoded caching with proactive routing, [45] proposes a solution in which groups of neighboring square-lets will cooperate together to form a *local group*. It is proved [45] that for a square local group with side length $s_g(n) = \Theta\left(\sqrt{\frac{m}{nM}}\right)$, any requested content is available in at least one node in the local group. A routing protocol [62] within the local group connects the source to destination through a series of horizontal and vertical square-lets. In this approach, it is assumed that a global knowledge of cached contents within each local group is known to all the nodes in that local group. Such an assumption results in significant overhead which requires allocating part of network capacity to the exchange of this information [75]. Further, exchange of this information poses significant security threat and allows an eavesdropper to find out which contents are cached in the local group.

In this paper, we propose a decentralized coded caching approach based on random vectors. In this technique the contents are randomly and independently combined and stored in caches. When a node requests a content, a unique linear combination of coded files can reconstruct the requested content. Therefore, each node first combines its encoded files and then transmits it to the anchor node (yellow circles in Figure 5.1). The anchor node also adds some information from its cache and forwards the newly updated file to the next anchor node closer to the requesting node as shown in the lower left local group in Figure 5.1. This process continues until the requesting node receives all the required coded files for decoding as shown in Figure 5.2. For reactive routing

approach, a random direction in the network is selected. Using the cached information in this random direction, the desired content can be obtained as shown in the upper left corner of Figure 5.1. Interestingly, we prove selecting a random direction is optimal in terms of number of hops traversed to retrieve the content. In both cases, perfect communication secrecy is achievable.

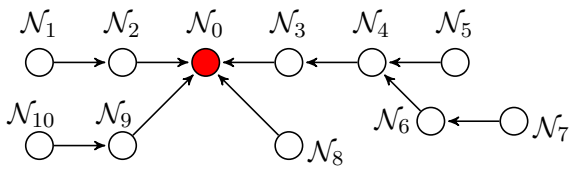

Figure 5.2: When a node $\mathcal{N}_0$ requests a file, it starts gathering the coded files from all the nodes in the local group. Once it gathers all these files, it adds its own coded files to it to create its desired file.

5.3.2 Decentralized Coded Cache Placement

In our proposed coded cache placement approach, a randomly encoded file $\mathbf{r}_j^i$ is created and placed in the jth cache location of node i. This randomly encoded file is a bit-wise summation of random contents from $\mathcal{F}$. In other words,

$$\mathbf{r}_j^i = \sum_{l=1}^{m} a_l^{i,j} F_l = \mathbf{v}_j^i \mathbf{F}, \tag{5.1}$$

where $\mathbf{F} = [F_1 \; F_2 \; \ldots \; F_m]^T$ represents the contents vector and $\mathbf{v}_j^i = [a_1^{i,j} \; a_2^{i,j} \; \ldots \; a_m^{i,j}]^T \in \mathbb{F}_2^m$ is a random row vector with each element equal to 0 or 1 and the summation is carried over Galois Field GF(2). This process is repeated independently for all cache locations of all nodes[4]. Notice that based on this construction, each vector $\mathbf{v}_j^i$ is uniformly selected

[4]In practice, each node chooses M linearly independent random vectors during cache placement phase. However, to simplify the analysis, we drop the independence assumption for different cache

from the set of all vectors in $\mathbb{F}_2^m$. Notice that this specific choice of fountain codes is known as Random Linear Fountain (RLF) codes [71] or Random Uniform LT codes [90].

5.3.3 Content Reconstruction

In order to decode any of the contents, nodes need to find m linearly independent vectors $\mathbf{v}_j^i$ to span the m-dimensional space. For each requested content F_r, a unique combination of these encoded files will generate F_r. In both routing scenarios, the computation of appropriate gains for the combination of coded files is carried by the requesting node and this information is relayed to the neighboring nodes.

As depicted in Figure 5.2, node $\mathcal{N}_i$ in the routing path can contribute up to M linearly independent row vectors $\mathbf{v}_1^i, \mathbf{v}_2^i, \ldots, \mathbf{v}_M^i$ to span the entire space. Node $\mathcal{N}_i$ applies gain $b_j^i \in \{0, 1\}$ to its jth cached file $\mathbf{r}_j^i$ and then adds (in binary field) $\sum_{j=1}^{M} b_j^i \mathbf{r}_j^i$ to the file it has received from previous hop and passes the newly constructed file to the next hop toward requesting node $\mathcal{N}_0$. After a total of N_r transmissions, the requesting node receives $(\sum_{i=1}^{N_r} \sum_{j=1}^{M} b_j^i \mathbf{v}_j^i)\mathbf{F}$. It will then applies decoding coefficients to its own cached files and adds it to the received file to reconstruct the desired content. Note that each relay node adds some encoded files to the received file and relays it forward. The coefficients $b_j^i \in \{0, 1\}$ are selected such that the linear combination of encoded files produce the desired requested content.

locations of a node. Therefore, analytical capacity results found in this paper are pessimistic and in practice the network capacity will increase.

5.3.4 Prior Results

The coded caching was originally introduced in [52] and [57] for cellular networks. Lemma 4.5.1 proved in [52] and [57] shows that the average number of cached files to reconstruct any content is equal to $m+c_3$ which is very close to m for large values of m. This shows that random uniform vectors perform close to optimal in terms of minimizing the number of cache locations. Based on Lemma 4.5.1, we have the following corollary.

Corollary 5.3.1. If random uniform vectors are used to create encoded files and then these files are independently cached in node caches, then on average $\mathbb{E}[N] = (m + c_3)/M = \Theta(m/M)$ nodes are required to decode any requested content.

Theorem 5.3.2. In the proposed coded caching scheme, selecting a random direction is asymptotically optimal in terms of minimizing the number of hops required to retrieve all the contents.

Proof. In the proposed coded caching scheme, random vectors in $\mathbb{F}_2^m$ are used for encoding the contents. In Lemma 4.5.1 which is proved in our earlier publications [52] and [57], we have shown that on average $m + c_3$ (where $c_3 \approx 1.6067$) random vectors are needed to span the whole m-dimensional space of $\mathbb{F}_2^m$. Therefore, with the proposed decentralized coded caching scheme on average $\mathbb{E}[N] = \lceil (m+c_3)/M \rceil$ nodes are required to decode any requested content. Therefore, no matter which routing direction is chosen for content retrieval, if on average $\mathbb{E}[N] = \lceil (m+c_3)/M \rceil$ hops are traversed, any content can be reconstructed.

On the other hand, to be able to retrieve all the contents, at least m cache locations are necessary. This means that at least $\lceil m/M \rceil$ nodes are required for content retrieval in any caching scheme. Since $m + c_3$ is very close to m for large values of m, then $\Theta(\lceil (m+c_3)/M \rceil) = \Theta(\lceil m/M \rceil) = \Theta(m/M)$, this proves that selecting any random direction is asymptotically optimal in terms of the minimum required number of hops for content retrieval. $\square$

Notice that Theorem 5.3.2 is intuitively valid since we have used completely random vectors and this means that all of the contents are equally distributed in all directions.

5.4 Capacity

This section is dedicated to computation of network throughput capacity of decentralized coded and uncoded caching schemes for proactive and reactive routing techniques. First, we define the precise notions of achievable throughput and network throughput capacity as follows.

Definition 5.4.1. A network throughput of $\lambda(n)$ contents per second for each node is *achievable* if there is a scheme for scheduling transmissions in the network, such that every content request by each node can be served by a rate of $\lambda(n)$ contents per second.

Whether a particular network throughput is achievable depends on the specific cache placement and node locations in the network. Since the location of the nodes and

the cache placement in nodes is random, we will define the network capacity as the maximum asymptotic network throughput achievable with probability 1.

Definition 5.4.2. We say that the *throughput capacity* of the network is lower bounded by $\Omega(g(n))$ contents per second if a deterministic constant $c_5 > 0$ exists such that

$$\lim_{n\to\infty} \mathbb{P}[\lambda(n) = c_5 g(n) \text{ is achievable }] = 1. \tag{5.2}$$

We say thet the network throughput capacity is upper bounded by $\mathrm{O}(g(n))$ contents per second if a deterministic constant $c_6 < +\infty$ exists such that

$$\liminf_{n\to\infty} \mathbb{P}[\lambda(n) = c_6 g(n) \text{ is achievable }] < 1. \tag{5.3}$$

We say that the network throughput capacity is of order $\Theta(g(n))$ contents per second if it is lower bounded by $\Omega(g(n))$ and upper bounded by $\mathrm{O}(g(n))$.

In this chapter, we study the throughput capacity after the cache placement phase and during the content delivery phase. In the following, we will describe the necessary size of the local group to decode all the contents in proactive routing approach.

Remark 5.4.3. Corollary 5.3.1 suggests that for decentralized coded caching on average $\Theta(m/M)$ nodes are needed to decode any desired content. Since any local group in proactive routing on average has $\Theta(ns_g(n)^2)$ nodes, then the average local group side length of $s_g(n) = \Theta\left(\sqrt{\frac{m}{nM}}\right)$ will be enough to decode all the contents. Notice that similar local group side length is found in [45] for the case of uncoded caching.

5.4.1 Capacity of proactive routing approach

In this section we assume that any node in each local group is completely aware of all the files cached in its local group. Nodes in local groups are cooperating with each other to transfer a requested content. In uncoded caching scenario it has been proved [45] that if $M \leq m < nM$, a capacity of $\Theta\left(\sqrt{M/m}\right)$ is achievable. Our proposed decentralized coded caching approach achieves a capacity of $\Theta\left(M/(m\log(n))\right)$ while providing perfect secrecy as will be proved subsequently.

Theorem 5.4.4. In decentralized coded caching with a proactive routing scheme the following network throughput capacity is achievable

$$\lambda(n) = \Theta\left(\frac{M}{m\log(n)}\right). \tag{5.4}$$

Proof. Corollary 5.3.1 shows that on average $\mathbb{E}[N] = \Theta(m/M)$ nodes are required to reconstruct all the contents in decentralized coded caching. Remark 5.4.3 shows that each local group of side length $\Theta\left(\sqrt{\frac{m}{nM}}\right)$ contains this many nodes. As shown in Figure 5.1, all these nodes cooperate to transmit the content to the requesting node. Since there are $\Theta(\log(n))$ nodes in each square-let, the total number of transmissions for one request in a local group is equal to $\Theta\left(\log(n)\left(\frac{s_g(n)}{s(n)}\right)^2\right)$. Therefore, the total number of file transmissions to satisfy all content requests in each local group has the order of $\Theta\left(\log(n)\left(\frac{s_g(n)}{s(n)}\right)^2\frac{m}{M}\right)$. On the other hand, in each local group we can have $\Theta\left(\frac{s_g(n)}{s(n)}\right)^2$

simultaneous transmissions. This implies that a network throughput of

$$\lambda(n) = \Theta\left(\frac{\left(\frac{s_g(n)}{s(n)}\right)^2}{\log(n)\left(\frac{s_g(n)}{s(n)}\right)^2 \frac{m}{M}}\right) = \Theta\left(\frac{M}{m\log(n)}\right) \tag{5.5}$$

is achievable. □

5.4.2 Capacity of reactive routing approach

Reactive routing usually requires less overhead but incurs higher delays in content delivery. However, one of the advantages of our proposed coded caching is that we can select any random direction and decode the desired content with the same optimum number of nodes. Such a scenario is depicted in the upper left corner of Figure 5.1.

For simplicity of our capacity analysis, we assume that all contents are of equal size with each having Q bits. This is a reasonable assumption since in practice the contents are divided into equal-sized chunks. Many references including [53, 72, 73] have assumed that the files are divided into equal chunks to be used during the cache placement phase. Assume that $\lambda(n)$ is the maximum achievable network throughput. This implies that with a probability close to one, the network can deliver $n\lambda(n)$ contents per second. Therefore, with a probability close to one, network nodes can transmit $n\lambda(n)\mathbb{E}[N]Q$ bits per second. There are exactly $\frac{1}{(c_2c_1s(n))^2}$ square-lets at any time slot available for transmission. Hence, the total number of bits that the network is capable of delivering is equal to $\frac{W}{(c_2c_1s(n))^2}$ where W is the total available bandwidth. Therefore,

$$\lambda(n) = \frac{W}{n\mathbb{E}[N]Q(c_2c_1s(n))^2} = \Theta\left(\frac{1}{\mathbb{E}[N]\log n}\right), \tag{5.6}$$

where W and Q are the total available bandwidth and total number of bits for each content respectively. This suggests that to find the maximum achievable network throughput, it is enough to find the average number of transmission hops needed to deliver the contents.

Theorem 5.4.5. In decentralized coded caching with reactive routing, the network capacity is

$$\lambda(n) = \Theta\left(\frac{M}{m\log(n)}\right). \tag{5.7}$$

Proof. Using Lemma 4.5.1, on average $\mathbb{E}[N] = \Theta(\frac{m}{M})$ nodes (or equivalently hops) are required to decode a content in decentralized coded caching. This along with equation (5.6) proves the theorem. □

In decentralized uncoded caching strategy, nodes cache contents with uniform distribution. We use Lemma 4.4.2 to find the average number of traveled hops in this case.

Theorem 5.4.6. If $m >> M$, the capacity of the network using decentralized uncoded caching is equal to

$$\lambda = \Theta\left(\frac{M}{m\log(m)\log(n)}\right). \tag{5.8}$$

Proof. Lemma 4.4.2 shows that in case of uncoded caching the average number of nodes

needed so that all of the requests can be satisfied is between

$$\frac{mH_m}{d(m,M)} \leq \mathbb{E}[N] \leq 1 + \frac{mH_m}{d(m,M)}. \tag{5.9}$$

Therefore, for large values of m, the average number of nodes for decoding scales as

$$\mathbb{E}[N] = \Theta\left(\frac{mH_m}{d(m,M)}\right). \tag{5.10}$$

To find a bound for $d(m,M)$, notice that the series in the right hand side of equation (4.9) has M terms and the maximum term $\frac{m}{m-M+1}$ corresponds to the case when $j = M-1$ and the minimum term 1 corresponds to the case when $j = 0$. A lower bound and an upper bound on $d(m,M)$ can be found by using the minimum and maximum terms respectively. Therefore,

$$M \leq d(m,M) \leq \frac{Mm}{m-M+1}. \tag{5.11}$$

When $m >> M$, then the lower and upper bounds of equation (5.11) converge to the same value of M.

$$\mathbb{E}[N] = \Theta\left(\frac{m\log(m)}{M}\right) \tag{5.12}$$

Combining (5.12) and (5.6) proves the theorem. □

Remark 5.4.7. Theorems 4.5.3 and 5.4.6 show that decentralized coded caching strategy can improve the network capacity by a factor of $\log(m)$ in reactive routing.

Figure 5.3 compares the capacity of coded caching with uncoded caching for both proactive and reactive routing algorithms. To plot this figure, we assumed that the

number of contents m, grows polynomially with the number of nodes n. Coded caching provides perfect secrecy as will be discussed later while it performs better (worse) than uncoded caching for reactive (proactive) routing algorithm.

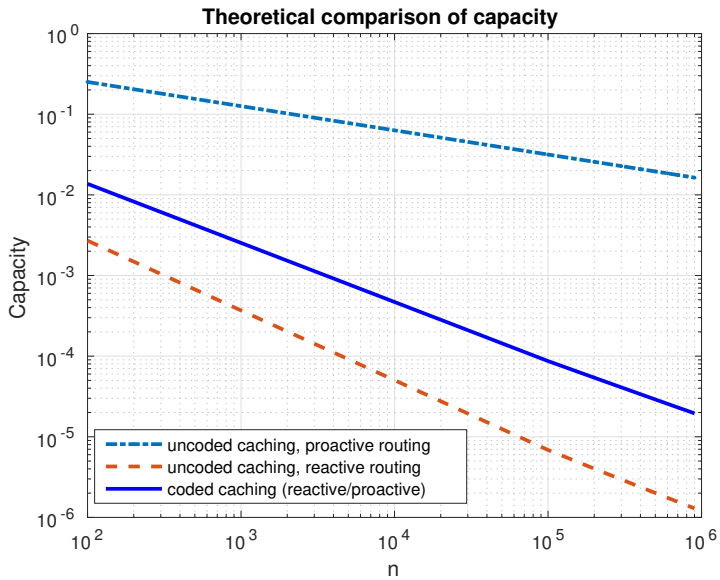

Figure 5.3: Capacity for coded and uncoded caching using proactive and reactive routing algorithms.

5.5 Security

This section evaluates the security of coded caching strategy. Note that uncoded caching allows an adversary with a noiseless wiretap channel to perfectly receive the transmitted files. We prove that such an eavesdropper will not be able to reduce its equivocation about the transmitted files in coded caching approach when there is a large number of files. Therefore, asymptotic perfect secrecy can be achieved. This problem was originally studied by Shannon [88].

Our secrecy proof is applicable to both proactive and reactive routing schemes. As described earlier, each node combines its encoded files as $\mathbf{x}^i = \sum_{j=1}^{M} b_j^i \mathbf{r}_j^i = \sum_{j=1}^{M} b_j^i \mathbf{v}_j^i \mathbf{F}$ and adds it to the previously received file and forwards it to the next hop. Therefore, the aggregate received file by the requesting node $\mathcal{N}_0$ is

$$S_r = \sum_{i=1}^{N_r} \mathbf{x}^i = \sum_{i=1}^{N_r} \sum_{j=1}^{M} b_j^i \mathbf{r}_j^i. \tag{5.13}$$

We assume that the encoding vectors $\mathbf{v}_j^i$ of the neighboring nodes are transmitted to the requesting node $\mathcal{N}_0$ through a secure low bandwidth channel. When enough number of such vectors are gathered, $\mathcal{N}_0$ computes the decoding coefficients b_j^i in order to generate the desired file F_r. Then, it sends back the b_j^i coefficients through the low bandwidth secure channel to its neighboring nodes. The secure channel is used only to transmit the encoding and decoding information. Transmitting the large encoded files through secure channel would be undesirable due to low bandwidth constraint. However, the encoding vectors $\mathbf{v}_j^i$ and the decoding gains b_j^i have much smaller sizes and they can be securely transmitted over the low bandwidth secure channel.

If content F_r is used in the encoding of at least one of the coded cached files in $\mathcal{N}_0$ (i.e. $a_r^{0,j} = 1$ for some $1 \leq j \leq M$), then the requesting node $\mathcal{N}_0$ can generate the file

$$\mathbf{x}_r^0 = \sum_{j=1}^{M} b_j^0 \mathbf{r}_j^0 = \sum_{j=1}^{M} b_j^0 \mathbf{v}_j^0 \mathbf{F} = F_r + \sum_{j=1}^{M} b_j'^0 \mathbf{v}_j^0 \mathbf{F} \tag{5.14}$$

from its own encoded cached files. Note that $\mathbf{v}_{\text{req}} = \sum_{j=1}^{M} b_j'^0 \mathbf{v}_j^0$ is a coding vector in m-dimensional space. Node $\mathcal{N}_0$ only needs to receive $S_r = \mathbf{v}_{\text{req}} \mathbf{F}$ and add S_r to $\mathbf{x}_r^0$ in GF(2) to retrieve F_r. The requesting node $\mathcal{N}_0$ uses the secure channel to collect

enough number of encoding vectors in order to span $\mathbf{v}_{\text{req}}$. Then the requesting node $\mathcal{N}_0$ finds the appropriate decoding coefficients b_j^i to span $\mathbf{v}_{\text{req}}$ and sends these decoding coefficients back to the neighboring nodes through the secure channel. The neighboring nodes collaboratively create the right decoding file S_r and transmit it to $\mathcal{N}_0$.

The second possibility which is less likely to happen for large values of M is that none of the encoded files in $\mathcal{N}_0$ contains F_r (i.e. $a_r^{0,j} = 0$ for all $1 \leq j \leq M$). In that case, $\mathcal{N}_0$ generates a unique combination of its encoded files as $\mathbf{x}_r^0 = \sum_{j=1}^{M} b_j^0 \mathbf{v}_j^0 \mathbf{F}$. In order to decode F_r, node $\mathcal{N}_0$ needs to receive $S_r = F_r + \mathbf{x}_r^0$. Hence, it uses the secure channel to collect enough number of encoding vectors $\mathbf{v}_j^i$ to be able to construct S_r. After solving the linear equation in GF(2), it sends the decoding gains back to the neighboring nodes such that they can collaboratively create the encoded file $S_r = F_r + \mathbf{x}_r^0$. When $\mathcal{N}_0$ receives S_r, it can add it to $\mathbf{x}_r^0$ to decode F_r.

We claim asymptotic perfect secrecy for this approach is achievable as long as for each requested file F_r, the requesting node generates a different encoded combination $\mathbf{x}_r^0$ which acts similar to key and hence, the transmitted signal is in fact an encrypted version of the message F_r. The intended receiver is indeed capable of decrypting the message by adding its secret key $\mathbf{x}_r^0$ to it. Notice that for each requested file F_r, a different key $\mathbf{x}_r^0$ is generated using the encoded cached files in the requesting nodes. No other eavesdropper will be able to decode the message as they do not have the secret key $\mathbf{x}_r^0$. The main advantage of the legitimate receiver is the information stored in its cache which allows it to create a unique key $\mathbf{x}_r^0$ for each requested file F_r. This is similar to Shannon cipher problem [88].

In [88], Shannon introduced the Shannon cipher system in which an encoding function $\mathfrak{e} : \mathbb{M} \times \mathbb{K} \rightarrow \mathbb{C}$ is mapping a message $\mathfrak{M}$ from the set of all messages $\mathbb{M}$ and a key $\mathfrak{K}$ from a set of keys $\mathbb{K}$ to a codeword $\mathfrak{C}$ from the set of codewords $\mathbb{C}$. In our problem, for each requested content by a user, the legitimate receiver uses a unique key $\mathfrak{K}$ to recover the message $\mathfrak{M}$. Even when a different user requests the same file, it uses a different key because each user caches different encoded files. The unique key for each user depends on the coded files that the node is storing and the coded files from neighboring nodes used for decoding the requested file. Shannon proved that if a coding scheme for Shannon's cipher system achieves perfect secrecy, then $\mathbb{H}(\mathfrak{K}) \geq \mathbb{H}(\mathfrak{M})$ where $\mathbb{H}(.)$ denotes the entropy. He proved that at least one secret key bit should be used for each message bit to achieve perfect secrecy. If the sizes of messages, keys and codewords are the same, there are necessary and sufficient conditions [11] to obtain perfect secrecy presented in the following theorem.

Theorem 5.5.1. If $|\mathbb{M}| = |\mathbb{K}| = |\mathbb{C}|$, a coding scheme achieves perfect secrecy if and only if

- For each pair $(\mathfrak{M}, \mathfrak{C}) \in (\mathbb{M} \times \mathbb{C})$, there exists a unique key $\mathfrak{K} \in \mathbb{K}$ such that $\mathfrak{C} = \mathfrak{e}(\mathfrak{M}, \mathfrak{K})$.
- The key $\mathfrak{K}$ is uniformly distributed in $\mathbb{K}$.

Proof. The proof can be found in section 3.1 of [11]. □

We will use Theorem 5.5.1 to prove that our approach can achieve asymptotic

perfect secrecy. In either of the cases, the requesting node $\mathcal{N}_0$ receives a codeword[5] $S_r = F_r + \mathbf{x}_r^0$ from the last node adjacent to $\mathcal{N}_0$. Node $\mathcal{N}_0$ uses XOR operation to decode F_r from S_r using it's secret key $\mathbf{x}_r^0$. Therefore, we have a Shannon cipher system in which $\mathfrak{M} = F_r, \mathfrak{K} = \mathbf{x}_r^0, \mathfrak{C} = S_r$ and $\mathfrak{e}$ denotes the XOR operation. To use this theorem, first we prove that for large enough values of m, the key $\mathbf{x}_r^0$ is uniformly distributed.

Lemma 5.5.2. The asymptotic distribution of bits of coded files in caches tend to uniform.

Proof. We assume that all files have Q bits and they may have a distribution different from uniform. We will prove that each coded cache file will be uniformly distributed for large values of m. Let us denote the kth bit of file F_l by f_l^k where $1 \leq k \leq Q$ and $1 \leq l \leq m$. Assume that $\Pr(f_l^k = 1) = p_l^k = 1 - \Pr(f_l^k = 0)$. Further, we assume that the bits of files (f_l^k) are independent. The kth bit of the coded file in the jth cache location of node i can be represented as

$$r_{i,j}^k = \sum_{l=1}^{m} a_l^{i,j} f_l^k, \tag{5.15}$$

where $a_l^{i,j}$ is a binary value with uniform distribution and independent of all other bits. Using regular summation (not over GF(2)) and denoting $h_l^{i,j,k} \triangleq a_l^{i,j} f_l^k$, we define $H^{i,j,k} \triangleq \sum_{l=1}^{m} h_l^{i,j,k}$. Therefore, $\Pr[r_{i,j}^k = 0] = \Pr[H^{i,j,k} \overset{2}{=} 0]$. Therefore, the kth bit of the coded file is equal to 0 if an even number of terms in $H^{i,j,k}$ is equal to 1. The probability distribution of $H^{i,j,k}$ can be computed using *probability generating functions.*

[5]This is true for both scenarios because all the operations are in GF(2).

Since $h_l^{i,j,k}$ is a Bernoulli random variable with probability $\frac{1}{2}p_l^k$, its probability generating function is equal to

$$G_l^{i,j,k}(z) = (1 - \frac{1}{2}p_l^k) + \frac{1}{2}p_l^k z. \tag{5.16}$$

Since $a_l^{i,j}$ and f_l^k are independent random variables, $h_l^{i,j,k}$ will become independent random variables. Therefore, the probability generating function of $H^{i,j,k}$ denoted by $G_H^{i,j,k}(z)$ is the product of all probability generating functions.

$$G_H^{i,j,k}(z) = \prod_{l=1}^{m} \left((1 - \frac{1}{2}p_l^k) + \frac{1}{2}p_l^k z \right). \tag{5.17}$$

Denoting the probability distribution of $H^{i,j,k}$ as $\mathfrak{h}(.)$, the probability of $H^{i,j,k}$ being even is

$$\begin{aligned}
&\Pr[H^{i,j,k} \overset{2}{\equiv} 0] = \sum_{u=0}^{\lfloor \frac{m}{2} \rfloor} \mathfrak{h}(2u) = \sum_{u=0}^{\lfloor \frac{m}{2} \rfloor} \mathfrak{h}(2u) z^{2u} \Big|_{z=1} \\
&= \frac{1}{2} \left[\sum_{u=0}^{m} \mathfrak{h}(u) z^u + \sum_{u=0}^{m} \mathfrak{h}(u)(-z)^u \right]_{z=1} \\
&= \frac{1}{2} G_H^{i,j,k}(1) + \frac{1}{2} G_H^{i,j,k}(-1) = \frac{1}{2} \prod_{l=1}^{m} \left((1 - \frac{1}{2}p_l^k) + \frac{1}{2}p_l^k \right) \\
&+ \frac{1}{2} \prod_{l=1}^{m} \left((1 - \frac{1}{2}p_l^k) - \frac{1}{2}p_l^k \right) = \frac{1}{2} \left(1 + \prod_{l=1}^{m} \left(1 - p_l^k \right) \right)
\end{aligned}$$

Therefore,

$$\begin{aligned}
&\lim_{m \to \infty} \Pr[r_{i,j}^k = 0] = \lim_{m \to \infty} \frac{1}{2} \left(1 + \prod_{l=1}^{m} \left(1 - p_l^k \right) \right) = \frac{1}{2} + \\
&\frac{1}{2} \lim_{m \to \infty} \prod_{l=1}^{m} \left(1 - p_l^k \right) = \frac{1}{2} + \frac{1}{2} \lim_{m \to \infty} \left(1 - \inf\{p_l^k\} \right)^m = \frac{1}{2}.
\end{aligned}$$

This proves the lemma. □

This lemma paves the way to prove the following theorem.

Theorem 5.5.3. The proposed coded caching strategy provides asymptotic perfect secrecy for the last hop if m is large and $m < 2^M$.

Proof. To formulate this as a Shannon cipher problem, we assume that $\mathfrak{M} = F_r$, $\mathfrak{K} = \mathbf{x}_r^0$, and $\mathfrak{C} = S_r$. The condition $m < 2^M$ ensures that a unique key exists for each requested message since at most 2^M possible random keys can be built from M cached files. The encoding function is XOR operation. For any pair $(\mathfrak{m}, \mathfrak{C}) \in (\mathbb{M}, \mathbb{C})$, a unique key $\mathfrak{K} \in \mathbb{K}$ exists such that $\mathfrak{C} = \mathfrak{m} + \mathfrak{K}$ which guarantees that $|\mathbb{M}| = |\mathbb{K}| = |\mathbb{C}|$.

Notice that the key $\mathfrak{K} = \mathbf{x}_r^0$ belongs to the set of all possible bit strings with Q bits. Lemma 5.5.2 proves that each coded cached content is uniformly distributed among all Q-bit strings. Hence each key which is a unique summation of cached encoded files is uniformly distributed among the set of all Q-bit strings. In other words, regardless of the distribution of the bits in files, $\mathbf{x}_r^0$ can be any bit string with equal probability for large values of m. Therefore, conditions of Theorem 5.5.1 are met and asymptotic perfect secrecy is achieved. □

Remark 5.5.4. In this chapter, we only studied the security of last hop communications in our approach and proved that even in the most vulnerable (last) link, secure communications is possible. A more general security study for all links remains as future work. Also, the study of the security of this approach against cooperative eavesdroppers remains as future work.

5.6 Cache Hit Probability

This section is dedicated to computation of cache hit probability when a node $\mathcal{N}_0$ can access u other nodes $\mathcal{N}_1, \ldots, \mathcal{N}_u$ or equivalently, $l = uM$ cache locations. We compute the event that $\mathcal{N}_0$ can decode any desired file in the set $\mathcal{F}$ with this information. First, let's define the cache hit probability.

Definition 5.6.1. The *cache hit probability for all contents* is defined as the probability that any content can be retrieved using the cached information in nodes $\mathcal{N}_1, \ldots, \mathcal{N}_u$.

We will first study uncoded cache hit probability.

5.6.1 Uncoded Caching

In uncoded caching, each node is randomly choosing M different contents from the set of m contents. This can be modeled as a coupon collector problem with group drawings in which a coupon collector is collecting a number of different coupons in each time and wants to find the probability that after u group collections, all the contents are collected. This problem is well-studied in [48, 91] and summarized in the following theorem.

Theorem 5.6.2. Assume that a coupon collector collects m different coupons. Each time a bundle of M different coupons are drawn uniformly at random. If u bundles are collected, then the probability that all the coupons are collected is equal to

$$\mathbb{P}_{\text{all collected}} = \sum_{j=0}^{m-M} (-1)^j \binom{m}{j} \left(\frac{\binom{m-j}{M}}{\binom{m}{M}} \right)^u \tag{5.18}$$

Proof. The proof can be found in [91] (a special case of Theorem 1) and also in page 164 of [48]. □

Result of Theorem 5.6.2 is the cache hit probability for collecting all contents when uncoded caching is used.

5.6.2 Coded caching

In coded caching, we want to compute the probability that there exists m linearly independent vectors within $l = uM$ random encoding vectors. If $m > l = uM$, this probability is clearly zero. Therefore, without loss of generality, we compute this probability for when $l = uM \geq m$. This problem has been studied in the literature [60] and the results are summarized below.

Theorem 5.6.3. Let $l \geq m \geq 1$ and s be positive integers and $r = l - m$. If $A = [a_{ij}]$ is an $l \times m$ matrix whose elements are independent binary uniform random variables and $\rho_m(l)$ is the rank of matrix A in GF(2), then if $m \to \infty$ we have

$$\mathbb{P}[\rho_m(l) = m - s] \to 2^{-s(s+r)} \prod_{i=s+1}^{\infty} \left(1 - \frac{1}{2^i}\right) \times \prod_{j=1}^{r+s} \left(1 - \frac{1}{2^j}\right)^{-1}, \tag{5.19}$$

where the last product equals 1 for $s + r = 0$.

Proof. This is Theorem 3.2.1 in page 126 of [60]. □

Corollary 5.6.4. Let $l \geq m$ and $A = [a_{ij}]$ be an $l \times m$ matrix whose elements are independent binary uniform random variables and $\rho_m(l)$ be the rank of matrix A in

GF(2). If $m \to \infty$, then

$$\mathbb{P}[\rho_m(l) = m] \to \prod_{i=l-m+1}^{\infty} \left(1 - \frac{1}{2^i}\right). \tag{5.20}$$

Equation (5.20) is the cache hit probability for coded caching approach.

Remark 5.6.5. The cache hit probability for coded caching very quickly approaches 1 if l is slightly larger than m. In fact, there is a very sharp transitioning of the probability from 0 to 1 in coded caching around the point $l = m$. However, in uncoded caching, l should be much larger than m in order for the cache hit probability to get close to 1 (see Figure 5.5).

Remark 5.6.6. This result demonstrates that coded caching scheme utilizes the cache space efficiently and avoids over-caching unlike uncoded caching approach.

5.7 Cache Update Algorithm

In this section, a caching update algorithm is described. Let's assume a new content F_{new} should replace another content F_k based on some replacement policy such as Least Recently Used (LRU) or Least Frequently Used (LFU) policy. The network controller uses a bit scrambling technique to create a file F'_{new} with uniform bit distribution from F_{new}. Such bit scrambling techniques are widely used in communication systems to give the transmitted data useful engineering properties [43]. Notice that the bit scrambling technique makes F'_{new} equivalent of temporary secret key with uniform

distribution. The network controller then generates $F_k + F'_{\text{new}}$ and broadcasts this file to the network nodes. When node N_i receives $F_k + F'_{\text{new}}$, it will add $F_k + F'_{\text{new}}$ to all of its cached encoded files $\mathbf{r}_j^i$ which contain F_k, i.e., all $\mathbf{r}_j^i$'s for which $a_k^{i,j} = 1$. In other words, if in the jth location of node i we have

$$\mathbf{r}_j^i = F_k + \sum_{\substack{l=1 \\ l \neq k}}^{m} a_l^{i,j} F_l, \tag{5.21}$$

then $F_k + F'_{\text{new}}$ will be added to $\mathbf{r}_j^i$. This replaces F_k with F'_{new} in all encoded files that contains F_k. If some cached encoded files does not contain F_k, then no action is required for those encoded files. Nodes can then decode F'_{new} using the same decoding gains as for F_k without any additional overhead. When F'_{new} is decoded, then a de-scrambling algorithm can be used to recover F_{new} from F'_{new}. A pseudocode representation of our caching update protocol is shown in Algorithm 2.

Notice that during the caching update phase none of the actual contents is transmitted and any eavesdropper would only receive encoded version of the files. Therefore, with this caching update technique, the cached contents can be updated securely.

5.8 Simulation

This section describes simulation results to verify that the proposed decentralized coded caching scheme can significantly reduce the average number of traveresed hops to retrieve the contents. We will also show through simulations that selecting a random direction for content retreival is optimal. Further, in this section we will use simulations to show that the cache hit probability of coded scheme is much higher than

Algorithm 2 Cache Update Algorithm

```
procedure CACHE UPDATE
    Find the content F_k that should be replaced.
    Scramble F_new to get F'_new with uniform bits.
    Encode the new content F'_new with F_k as F'_new ⊕ F_k.
    Broadcast F'_new ⊕ F_k to all the nodes.
    for the j^th cache location of node i do
        if F_k is used in encoding r^i_j (i.e. a^{i,j}_k = 1) then
            Update the j^th cache location of node i with
            F'_new ⊕ F_k ⊕ r^i_j.
```

that of uncoded caching.

The simulation results in Figure 5.4 compare the performance of the proposed decentralized coded caching scheme with decentralized uncoded caching. We considered a wireless ad hoc network of $n = 1000$ nodes with $m = 100$ contents. In this figure, the average number of hops required to decode the contents in both decentralized coded and uncoded caching schemes are compared. The simulation results clearly demonstrate that decentralized coded caching outperforms uncoded case particularly when the cache size is small which is the usual operating regime. For instance, with decentralized coded content caching, a cache of size 25 only requires less than 5 hops while decentralized uncoded content caching needs around 20 hops for successful content retreival. This makes coded content caching much more practical compared to uncoded content caching.

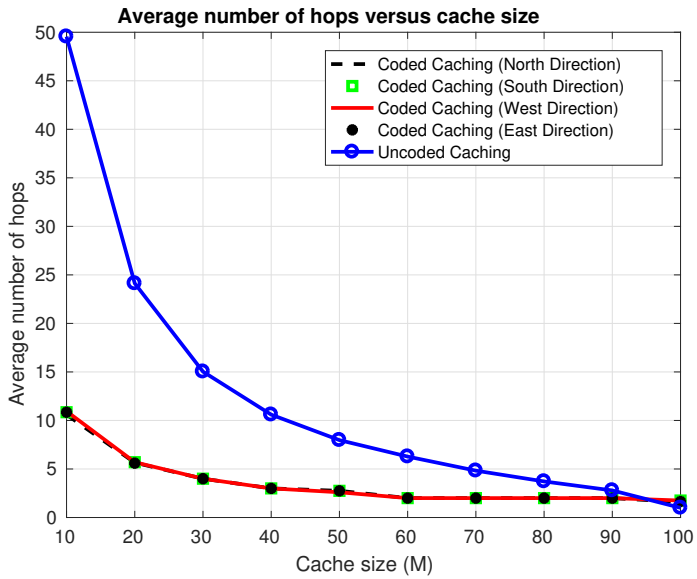

Figure 5.4: Simulation results show that the proposed coded caching approach can significantly reduce the traveresed number of hops when a node wants to access the contents.

Note that capacity is inversely proportional to the average hop counts. As can be seen from Figure 5.4, for small cache sizes, coded caching significantly reduces the number of hops required to decode the contents. This property is important for nodes with small storage capability since large number of hops can impose excessive delay and low quality of service.

Figure 5.4 on the other hand proves another important result that content retreival can be optimally done in any random direction. In this plot, we have used random directions of east, west, south and north for content retreival using coded caching and we have shown that the average number of hops in any of these random directions is the same. The four plots corresponding to these four directions is so close that it is hard to notice them at first sight.

Figure 5.5 shows the simulation results for different values of M when $m = 100$. The cache hit probability is plotted as a function of the number of cached contents l and for different cache sizes M. However, for each fixed value of l, the number of nodes $\mu = \frac{l}{M}$ will be different depending on M. For instance, at $l = 400$, M and μ are 25 and 16 respectively. The simulation results are validating the theoretical results in Theorem 5.6.2 and Corollary 5.6.4. As can be seen from this figure, the cache hit probability for coded caching is much higher than that of uncoded caching. Also, it is clear from this plot that the cache hit probability of coded caching approaches 1 rapidly when l starts to increase from m. For uncoded caching, specially when m is large, the receiver should access a much larger number of cached contents such that the cache hit probability approaches 1. Therefore, in networks with a large number of contents our coded caching approach will quickly achieve a cache hit probability of close to one with a much smaller number of cache locations. This is a significant benefit of our technique in reducing the overcaching.

5.9 Discussion

While in this chapter we considered a static network in which the location of nodes does not change, our results can be extended to dynamic mobile or vehicular networks in which the transmission time is much smaller than the required time to change the location of the vehicles. In future vehicular wireless networks where vehicles use millimeter waves to communicate, bandwidths of up to 60 GHz and data rates

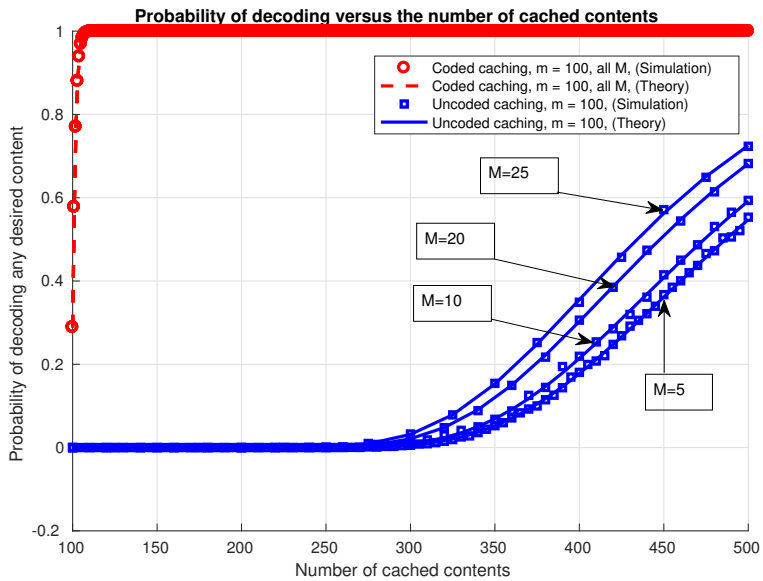

Figure 5.5: Cache hit probability for any desired content when $m = 100$.

of up to 700 Mpbs are achievable [94]. Large chunks of files can be downloaded in a very short time before the vehicles can move relative to each other in such networks. Further, the size of the file chunks can be adjusted based on the network dynamism. An adaptive file segementation algorithm can choose a low file chunk size for high changing network environments while it can choose a much larger file chunk size for slow changing networks.

Our goal in this chapter was to prove that we can achieve better security, cache hit probability and capacity results with coded caching. We considered the case when completely random vectors are uniformly selected to encode the contents. Our results in this chapter including the capacity and cache hit probability are based on this specific cache placement strategy. While other fountain coding choices for cache placement

may reduce the decoding complexity, they can also reduce the capacity or cache hit probability. Network designers should consider many factors including the decoding complexity, delay, number of hops and routing (reactive or proactive) for selecting the appropriate fountain coding scheme for cache placement.

Chapter 6

Fountain Coding Based Storage in Distributed Cloud Systems

In this chapter an information theoretic approach to security and privacy is introduced. The approach called Secure And Private Information Retrieval (SAPIR) is applied to distributed data storage systems. In this approach, random combinations of all contents are stored across the network. Our coding approach is based on Random Linear Fountain (RLF) codes. To retrieve a content, a group of servers collaborate with each other to form a Reconstruction Group (RG). SAPIR achieves asymptotic perfect secrecy if at least one of the servers within an RG is not compromised. Further, a private information retrieval scheme based on random queries is proposed. This information theoretic approach ensures the users to privately download their desired contents without the servers knowing about the requested contents indices. The proposed scheme is adaptive and can provide privacy against a significant number of colluding servers. The

results in this chapter are published in [54].

6.1 Motivation

Cloud networks have become a popular platform for data storage during the past decade. Security of the stored data has always been a major concern for many cloud service providers. Many cloud service providers use encryption algorithms to encrypt the data on their servers. Dropbox, for instance, is using Advanced Encryption Standard (AES) to store the contents on its servers[1]. Since the encryption algorithms are *computationally secure*, an adversary may be able to break them with time. For instance, Data Encryption Standard (DES) which was once the official Federal Information Processing Standard (FIPS) in US is not considered secure anymore. An interesting problem in highly sensitive cloud services would then be to look for *information theoretic secure* solutions which are immune to attackers in time.

To achieve perfect information theoretic secrecy using Shannon cipher system [88], the number of keys should be equal to the number of messages. Therefore, to retrieve the contents from the cloud using an information theoretically secure approach in which the contents are directly encoded with a different key, each user needs to store a huge number of keys which is not practical. In this chapter we propose a technique in which the storage capability of the trusted servers are efficiently used to generate the keys by using the contents themselves and achieve asymptotic perfect secrecy. Our proposed technique is based on Random Linear Fountain (RLF) codes [71]. RLF codes

[1] `https://www.dropbox.com/en/help/27`

have been shown [52, 56, 57] to be very useful in distributed storage systems.

On the other hand, in many distributed storage applications like Peer-to-Peer (P2P) distributed storage systems or distributed storage systems in which some of the servers are under the control of an oppressive government, a user wants to download a content from a pool of distributed servers in a way that the servers cannot determine which content is requested by the user. This is widely known as Private Information Retrieval (PIR) problem.

In this chapter, we propose a novel technique to address the PIR problem in distributed storage systems. In our solution, users use random queries to request data from the servers. These random queries are designed in a way that they can be used to retrieve any desired content while prevent any malicious agent with the knowledge of up to half of the random queries to gain information about the requested content. This is an important feature of the proposed technique that provides privacy in the presence of many colluding servers. Such a feature has not been presented in prior information theoretic PIR approaches [92] for coded storage systems. Our proposed Secure And Private Information Retrieval (SAPIR) scheme provides both security and privacy for information retrieval.

The rest of the chapter is organized as follows. Section 6.2 is dedicated to the related work on PIR and security in distributed storage systems. The assumptions and problem formulation are described in section 6.3. We study the security and PIR aspects of SAPIR in sections 6.4 and 6.5, respectively. The simulation results are provided in section 6.6.

6.2 Related Works

In this chapter, we use *Random Linear Fountain (RLF) codes* [71] to encode the contents within the servers in the network. Significant capacity improvement using RLF codes in wireless ad hoc and cellular networks has been shown in [39,40,51–53,56–58,75]. The application of fountain codes in distributed storage systems was also studied in [28].

In [56], we have computed the capacity of wireless ad hoc networks with caching and shown that RLF codes can be used to achieve perfect secrecy. However, we did not study the problem of PIR. We will use RLF codes to simultaneously achieve both security and privacy in distributed cloud applications.

While MDS codes [27, 29] show good repair capability, these codes are not particularly designed to provide security. Authors in [26] have studied the security of distributed storage systems with MDS codes and [63] has proposed a construction for repairable and secure fountain codes. Reference [63] achieves security by concatenating Gabidulin codes with Repairable Fountain Codes (RFC). Their specific design allows to use Locally Repairable Fountain Codes for secure repair of the lost data. Unlike [63] which has focused on the security of the repair links using concatenated codes, we present simultaneous security and privacy of the data storage nodes by only using RLF codes. References [95] and [97] have studied the problem of security in the presence of overhearing interference in cooperative communications. Further [76, 77] studied the same problem on multi-tier networks.

The authors in [24] have numerically studied the wiretap network with a simple

topology in which there is a relaying node between the transmitter and the receiver. We considered the general network with a cloud infrastructure in which the servers are cooperating to reconstruct the contents.

The idea of PIR was originally introduced in [25] for uncoded databases. Recently, there has been a renewed interest in studying PIR for storage systems utilizing different coding techniques. Reference [86] was among the first references to study the problem of PIR for coded storage systems. They proved that with only one extra bit, PIR can be achieved. However, the solution in [86] requires that the number of servers grows with the data record size. Reference [19] assumed that the number of servers is fixed and established the trade-off between storage and retrieval costs and demonstrated the fundamental limits on the cost of PIR for coded storage systems. The authors in [92] studied the problem of PIR for MDS coded storage systems and introduced a scheme to achieve PIR in MDS coded databases but the security aspect was not addressed in this chapter. They have also assumed that the databases are able to store all the contents which may not be a realistic assumption. Unlike prior work [19, 86, 92] which have studied PIR for coded databases, we are interested in achieving simultaneous security and PIR. Further, as far as we know, this is the first work to study the problem of PIR for a fountain coded-based distributed storage system. The proposed PIR scheme is easily scalable to the cases when up to half of the servers are colluding to obtain information about the content or content index which makes this technique very robust against large number of colluding servers.

6.3 Problem Formulation

The network is composed of n servers each capable of storing h contents. These servers are denoted by $\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_n$. A total number of m contents exist within the network and each content has M bits, i.e., $f_1, f_2, \ldots, f_m$.

6.3.1 RLF Coding-Based Storage

The contents are randomly encoded and stored on the servers during the data preloading phase. The encoded file in the jth storage location of the ith server for any $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, h$ will have the form

$$c_j^i = \sum_{k=0}^{m} v_k^{i,j} f_k = \mathbf{f}\mathbf{v}_j^i, \tag{6.1}$$

where[2] $\mathbf{f} = [f_1 \ f_2 \ \ldots f_m]$ denotes the $1 \times m$ vector of all contents and $\mathbf{v}_j^i$ denotes an $m \times 1$ random encoding vector of 0's and 1's. Each content f_i belongs to the Galois Field $\mathbb{F}_{2^M}$, i.e. $\mathbf{f} \in \mathbb{F}_{2^M}^m$. Unless otherwise stated we assume that all the vector and matrix operations are in $\mathbb{F}_2$. The encoded files stored in server $\mathcal{N}_i$ are $\mathbf{c}_i = [c_1^i \ c_2^i \ \ldots c_h^i]$ where $\mathbf{c}_i \in \mathbb{F}_{2^M}^h$. Note that $\mathbf{c}_i = \mathbf{f}\mathbf{V}_i$ where $\mathbf{V}_i$ is the $m \times h$ random encoding matrix for server $\mathcal{N}_i$.

In RLF all random vectors $\mathbf{v}_j^i$ are chosen independently and uniformly from $\mathbb{F}_2^m$ which results in a random uniform choice of the encoding matrix $\mathbf{V}_i$ where each element can be either 0 or 1 with equal probability. Such an encoding matrix may not necessarily be full rank and may contain linearly dependent rows. This will result in redundant

[2]Throughout this chapter, the vectors are denoted in bold characters.

use of storage and may jeopardize the security by revealing more information. Hence, we propose a *full rank encoding* scheme based on RLF codes in which randomly created encoding vectors $\mathbf{v}_j^i$ are discarded if they already exist in the span of the previously selected random encoding vectors. In other words, for each server we select h linearly independent vectors to construct a full rank matrix $\mathbf{V}_i$ of size $m \times h$ for $i = 1, 2, \ldots, n$.

The encoding can be performed in a decentralized way. This means that each server can fill up its storage space independently of all the other servers during the data preloading phase. It can be shown [57] that the average minimum number of encoded files required to decode any desired content is very close to the optimal value of m.

6.3.2 Reconstruction Groups (RG)

After the data preloading phase, users can reconstruct their desired contents during content delivery phase. A desired file f_r can be written as $f_r = \mathbf{f}\mathbf{e}_r$, where $\mathbf{e}_r$ is a all zero vector of size $m \times 1$ except in the rth location is equal to 1. To retrieve f_r, the user needs to access enough encoded files on the network servers in order to construct $\mathbf{e}_r$ via $\mathbf{v}_j^i$'s.

Since codes are constructed in $\mathbb{F}_2^m$, users need m linearly independent encoding vectors to retrieve any of the m contents. We assume that servers are divided into many different *RGs* . Servers within each RG collaborate with each other to retrieve any requested content. Therefore, the number of encoded files within a single RG should be at least equal to m. The RGs are represented by $\mathcal{J}_1, \mathcal{J}_2, \ldots, \mathcal{J}_u$ and the number of servers within their corresponding RGs by $J_1, J_2, \ldots, J_u$ where, $\sum_{i=1}^{u} J_i = n$. It is

shown in [57] that the average minimum number of encoded files within each RG to retrieve all the contents is only slightly larger than m. Therefore, for each RG $\mathcal{J}_i$ where $1 \leq i \leq u$, the minimum value of J_i is only slightly larger than $\frac{m}{h}$. Notice that if J_i is smaller than $\frac{m}{h}$, then the servers will not be able to form a full rank matrix to retrieve all desired contents. In the case that storage systems store uncoded contents, we need exactly m cache locations for storing files which is very close to our RLF technique and demonstrates that our RLF-coding based approach efficiently utilizes storage space. For large values of h, i.e. $h \geq m$, each server can become an RG by itself.

6.3.3 Content Retrieval

Each RG $\mathcal{J}_k$ stores $J_k h \geq m$ randomly encoded files. The matrices $\mathbf{V}_i$ of the J_k servers in the RG form a full rank matrix $\mathbf{V} = [\mathbf{V}_1 \ \mathbf{V}_2 \ \ldots \ \mathbf{V}_{J_k}]_{m \times J_k h}$. Therefore, any content with index r can be retrieved from the servers by solving the linear equation $\mathbf{V}\mathbf{y}_r = \mathbf{e}_r$ in $\mathbb{F}_2$. Since this matrix is full rank, one possible solution can be given as

$$\mathbf{y}_r = \mathbf{V}^T \left(\mathbf{V}\mathbf{V}^T\right)^{-1} \mathbf{e}_r. \tag{6.2}$$

To solve $\mathbf{V}\mathbf{y}_r = \mathbf{e}_r$, servers within the RG should send their corresponding encoding matrices $\mathbf{V}_i$ to one of the RG servers called $\mathcal{N}_s$ that generates $\mathbf{V}$ and computes $\mathbf{y}_r$ from the above equation[3]. If $\mathbf{y}_r = [\mathbf{y}_r^1 \ \mathbf{y}_r^2 \ \ldots \ \mathbf{y}_r^{J_k}]^T$ is such a solution, where $\mathbf{y}_r^i$ is a $h \times 1$ local decoding vector for server $\mathcal{N}_i$, then server $\mathcal{N}_s$ sends $\mathbf{y}_r^i$ to server $\mathcal{N}_i$ and $\mathcal{N}_i$ then transmits $\mathbf{f}\mathbf{V}_i\mathbf{y}_r^i$ to the requesting user. All of the server responses are then aggregated

[3]Notice that the servers of an RG only need to send this information to $\mathcal{N}_s$ once. This could be done even right after the data preloading phase.

by the user to retrieve f_r as

$$f_r = \mathbf{f}\mathbf{e}_r = \mathbf{f}\mathbf{V}\mathbf{y}_r = \sum_{i=1}^{J_k} \mathbf{f}\mathbf{V}_i \mathbf{y}_r^i. \tag{6.3}$$

However, this solution reveals the identity of the downloaded content to all the servers of the RG. This simple solution cannot be used for PIR but we will show in section 6.4 that perfect secrecy can be achieved with this solution. A solution to preserve the privacy of the users is presented in section 6.5.

6.4 Security

This section is dedicated to the study of security of our approach. If an adversary is able to wiretap all of the communication links between the RG servers and the user, it can perfectly retrieve f_r using equation (6.3). We prove that perfect communication secrecy can be achieved when the adversary can wiretap all communication links between servers and user except one. We will prove this for the case when the user directly sends the request $\mathbf{e}_r$ to the servers and the servers respond accordingly. Under this scenario, the adversary knows the requested content index but still unable to reduce its equivocation about the requested content.

Consider RG $\mathcal{J}_k$ and without loss of generality, assume that an adversary can wiretap all of the links between servers $\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_{J_k-1}$ and the user. Further assume that the user wants to directly download the content f_r from these servers by sending the query $\mathbf{e}_r$ to all these servers. Such a scenario is much more vulnerable to adversarial attacks compared to a scenario in which the requested base vectors are expanded in

terms of random queries in order to guarantee privacy. When the query $\mathbf{e}_r$ is received by all the servers,they will collectively solve the linear equation $\mathbf{V}\mathbf{y}_r = \mathbf{e}_r$ to find the decoding vector $\mathbf{y}_r$. Equation (6.3) can be rewritten as

$$f_r = \mathbf{f}\mathbf{e}_r = \mathbf{f}\mathbf{V}\mathbf{y}_r = \sum_{i=1}^{J_k-1} \mathbf{f}\mathbf{V}_i\mathbf{y}_r^i + \mathbf{f}\mathbf{V}_{J_k}\mathbf{y}_r^{J_k}. \tag{6.4}$$

Since we assume that all of the responses from the servers $\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_{J_k-1}$ can be wiretapped, we can assume that the first part of the above equation is known while the second part is secret to the adversary. Lets define $S_r \triangleq \sum_{i=1}^{J_k-1} \mathbf{f}\mathbf{V}_i\mathbf{y}_r^i$ and $T_r \triangleq \mathbf{f}\mathbf{V}_{J_k}\mathbf{y}_r^{J_k}$. The requested content can be written as $f_r = S_r + T_r$ and since all operations are in $\mathbb{F}_2$, we have

$$S_r = f_r + T_r. \tag{6.5}$$

This is similar to the Shannon cipher system [88] in which an encoding function $\mathfrak{e} : \mathbb{M} \times \mathbb{K} \to \mathbb{C}$ is mapping a message $\mathfrak{M} \in \mathbb{M}$ and a key $\mathfrak{K} \in \mathbb{K}$ to a codeword $\mathfrak{C} \in \mathbb{C}$. In our problem f_r, T_r, and S_r can be regarded as the message, key, and codeword respectively. The eavesdropper knows the encoded file S_r but it cannot obtain any information about the message f_r if a unique key T_r with uniform distribution is used for each message. Theorem 5.5.1 provides the necessary and sufficient condition [11] to obtain perfect secrecy. We will use Theorem 5.5.1 to prove that our approach can achieve asymptotic perfect secrecy. To use this theorem, first we prove that for large enough values of m, the key T_r is uniformly distributed.

Lemma 6.4.1. The asymptotic distribution of bits of coded files on the servers tend to uniform.

Proof. The proof is similar to the proof of Lemma 5.5.2. □

This lemma paves the way to prove the following theorem.

Theorem 6.4.2. For the proposed full rank encoding scheme if m is large but $m < 2^h$, then the proposed encoded strategy provides asymptotic perfect secrecy against any eavesdropper which is capable of wiretapping all but one of the links from the servers to a user in a RG.

Proof. We formulated this problem as a Shannon cipher system assuming that $\mathfrak{M} = f_r$, $\mathfrak{K} = T_r$, and $\mathfrak{C} = S_r$. The condition $m < 2^h$ ensures that a unique vector $\mathbf{y}_r^{J_k}$ exists for each requested message. Therefore, since full rank encoding scheme is used, then $\mathbf{V}_{J_k}$ will be full rank and T_r guarantees that a unique key exists for each requested message f_r. Notice that if the size of the RG is large enough, then the unique choice of the key does not affect the solvability of the linear equation $\mathbf{V}\mathbf{y}_r = \mathbf{e}_r$. Therefore, for any pair $(\mathfrak{m}, \mathfrak{C}) \in (\mathbb{M}, \mathbb{C})$, a unique key $\mathfrak{K} \in \mathbb{K}$ exists such that $\mathfrak{C} = \mathfrak{m} + \mathfrak{K}$. Further, we are guaranteed to have $|\mathbb{M}| = |\mathbb{K}| = |\mathbb{C}|$.

Notice that the key $\mathfrak{K} = T_r$ belongs to the set of all possible bit strings with M bits. Lemma 6.4.1 proves that each encoded file is uniformly distributed among all M-bit strings. Hence each key which is a unique summation of such encoded files is uniformly distributed among the set of all M-bit strings. In other words, regardless of the distribution of the bits in files, T_r can be any bit string with equal probability for large values of m. Therefore, the conditions in Theorem 5.5.1 are met and perfect secrecy is achieved. □

Remark 6.4.3. In this chapter, we have assumed that the decoding vector $\mathbf{y}_r$ and the encoding matrix $\mathbf{V}_i$ are computed during the data preloading phase securely. Therefore, the eavesdropper cannot decode this information on any of the servers or have any knowledge about the key T_r.

Remark 6.4.4. A naive approach to achieve perfect secrecy using the Shannon cipher system is to choose m different keys from the set of uniform M-bit strings and store them and use them to encode the files. However, since the file size M is very large, this requires a significant amount of storage space to store the keys on the trusted servers which doubles the required storage capacity. The important contribution of our approach is that users do not need to store the keys and yet perfect secrecy can still be achieved with the help of trusted servers.

6.5 Private Information Retrieval

In PIR, the goal is to provide conditions that when a user downloads the content f_r with index $r \in \{1, 2, \ldots, m\}$, the content index remains a secret to all of the servers. This is desirable in applications like Peer-to-Peer networks and in situations where some servers may have been compromised by the adversary. To achieve PIR, users send queries to the servers and servers respond to users based on those queries. These queries should be designed in a way that reveal no information to the servers about the requested content index. To formally define the *information theoretic PIR*, let R be a random variable denoting the requested content index and let $\mathcal{Q}_l$ be a subset

of at most l queries. We have the following definition.

Definition 6.5.1. A PIR scheme is capable of achieving perfect information theoretic PIR against i colluding servers if for the set $\mathcal{Q}_l$ of all queries available to all of these servers and any number of contents we have

$$I(R; \mathcal{Q}_l) = 0 \tag{6.6}$$

where $I(.)$ is the mutual information function.

6.5.1 Random Query Generation

To achieve PIR, the user chooses a fixed $\epsilon > 0$ and then sets $A^\epsilon \triangleq m + \lceil \log_2(\frac{1}{\epsilon}) \rceil$. Then it picks A^ϵ query vectors from $\mathbb{F}_2^m$ uniformly at random and statistically independent of each other. These will be the set of random queries. Therefore, we will have a set $\mathcal{Q}^\epsilon = \{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_{A^\epsilon}\}$ of i.i.d. random query vectors. In the following, we will prove that with a probability of at least $1 - \epsilon$, these random vectors span the whole m-dimensional space of $\mathbb{F}_2^m$. The properties of random vectors that we have used for our coding technique, had been previously studied in [60].

Theorem 6.5.2. Let $\mathbf{Q}$ be a matrix of size $m \times l$ whose elements are independent random variables taking the values 0 and 1 with equal probability and let $\rho_m(l)$ be the rank of the matrix $\mathbf{Q}$ in $\mathbb{F}_2$. Let $s \geq 0$ and c be fixed integers, $c + s \geq 0$. If $m \to \infty$ and $l = m + c$, then

$$\mathbb{P}[\rho_m(l) = m - s]$$

$$\to 2^{-s(s+c)} \prod_{i=s+1}^{\infty} \left(1 - \frac{1}{2^i}\right) \prod_{j=1}^{s+c} \left(1 - \frac{1}{2^j}\right)^{-1} \tag{6.7}$$

where the last product equals 1 for $c + s = 0$.

Proof. This is Theorem 3.2.1 in page 126 of [60]. □

Corollary 6.5.3. For $l = m + c$ where $c \geq 0$, if $m \to \infty$ we have

$$\mathbb{P}[\rho(l) = m] \to \prod_{i=c+1}^{\infty} \left(1 - \frac{1}{2^i}\right) \tag{6.8}$$

Proof. The proof follows for $s = 0$ in Theorem 6.5.2. □

In the following, we will use these results for our proofs.

Definition 6.5.4. We define the random variable A as the minimum number of random query vectors $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_i$ to span the whole space of $\mathbb{F}_2^m$.

Lemma 6.5.5. The probability of the event that $A < m$ is zero and for any $c \geq 0$ we have

$$\mathbb{P}[A \leq m + c] \to \prod_{i=c+1}^{\infty} \left(1 - \frac{1}{2^i}\right) \tag{6.9}$$

Proof. This is a direct result of Corollary 6.5.3. □

Lemma 6.5.6. The probability of the event that $A = m + c$ is less than 2^{-c} for any $c \geq 0$.

Proof. Let $F(c) \triangleq \mathbb{P}[A \leq m+c]$. It is easy to verify from equation (6.9) that for $m \to \infty$ we have

$$F(c) \to \frac{F(c-1)}{1-\frac{1}{2^c}}. \tag{6.10}$$

Since $F(c) \leq 1$, from equation (6.10) we arrive at

$$F(c-1) \leq 1 - 2^{-c}. \tag{6.11}$$

Hence,

$$\begin{aligned}\mathbb{P}[A = m+c] &= F(c) - F(c-1) \to F(c-1)\left(\frac{1}{1-\frac{1}{2^c}} - 1\right) \\ &= F(c-1)\left(\frac{1}{1-\frac{1}{2^c}} - 1\right) = \frac{F(c-1)}{2^c - 1} \leq \frac{1-2^{-c}}{2^c-1} = 2^{-c}\end{aligned}$$

□

Lemma 6.5.7. The probability of the event that $A \leq m+c$ is at least $1-2^{-c}$ and at most $1-2^{-(c+1)}$ for any $c \geq 0$. i.e.

$$1 - 2^{-c} \leq F(c) \leq 1 - 2^{-(c+1)} \tag{6.12}$$

Proof. The upper bound is already proved in equation (6.11). From Lemma 6.5.6 we have,

$$\begin{aligned}F(c) &= \mathbb{P}[A \leq m+c] = 1 - \mathbb{P}[A > m+c] \\ &= 1 - \sum_{i=c+1}^{\infty} \mathbb{P}[m+i] \geq 1 - \sum_{i=c+1}^{\infty} 2^{-i} = 1 - 2^{-c}\end{aligned}$$

□

Theorem 6.5.8. With a probability of at least $1-\epsilon$, the set of random queries $\mathcal{Q}^\epsilon = \{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_{A^\epsilon}\}$ where $A^\epsilon = m + \lceil \log_2(\frac{1}{\epsilon}) \rceil$ spans the whole m-dimensional space of $\mathbb{F}_2^m$.

Proof. From Lemma 6.5.7, we have

$$\mathbb{P}[A \leq A^\epsilon = m + \lceil \log_2(\frac{1}{\epsilon}) \rceil] \geq 1 - 2^{-\lceil \log_2(\frac{1}{\epsilon}) \rceil} \geq 1 - \epsilon$$

This proves the theorem. □

Theorem 6.5.8 states that the probability of spanning the m-dimensional space can arbitrarily go to 1 provided that the number of random vectors increases logarithmically with $\frac{1}{\epsilon}$. For example, to span the m-dimensional space with a probability of at least 0.99, it is enough to only have $m+7$ random vectors. Using these random query vectors, we can now show that even with a large number of colluding servers no information about the requested content index can be obtained. To prove this result, we need to prove some lemmas.

Let $\mathbf{Q}^\epsilon \triangleq [\mathbf{q}_1 \ \mathbf{q}_2 \ \ldots \ \mathbf{q}_{A^\epsilon}]$ be the matrix of size $m \times A^\epsilon$ whose columns are random query vectors. Matrix $\mathbf{Q}^\epsilon$ contains A^ϵ statistically independent random vectors. Let $B_{\mathbf{x}}^r$ be the event that for a specific vector $\mathbf{x} \in \mathbb{F}_2^{A^\epsilon}$ and a specific base vector $\mathbf{e}_r$, we have $\mathbf{Q}^\epsilon \mathbf{x} = \mathbf{e}_r$.

Lemma 6.5.9. For any specific non-zero vector $\mathbf{x} \in \mathbb{F}_2^{A^\epsilon}$ we have

$$\mathbb{P}[B_{\mathbf{x}}^r] = \mathbb{P}[\mathbf{Q}^\epsilon \mathbf{x} = \mathbf{e}_r] = 2^{-m}. \tag{6.13}$$

Proof. lets assume vector $\mathbf{x}$ has k ones. If $\mathbf{Q}^\epsilon\mathbf{x} = \mathbf{e}_r$, then k vectors from the set of all vectors $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_{A^\epsilon}$ are added together to create $\mathbf{e}_r$. Let's denote these vectors by $\mathbf{q}_{e_1}, \mathbf{q}_{e_2}, \ldots, \mathbf{q}_{e_k}$. Let $q_r^{e_j}$ denotes the rth element of vector $\mathbf{q}_{e_j}$. Since the vectors $\mathbf{q}_{e_1}, \mathbf{q}_{e_2}, \ldots, \mathbf{q}_{e_k}$ are independent and their elements are also mutually independent, using binary summations in $\mathbb{F}_2$ we have

$$\mathbb{P}[B_{\mathbf{x}}^r] = \mathbb{P}[\mathbf{Q}^\epsilon\mathbf{x} = \mathbf{e}_r] = \mathbb{P}[\sum_{j=1}^{k} q_r^{e_j} = 1] \prod_{\substack{l'=1 \\ l' \neq r}}^{m} \mathbb{P}[\sum_{j=1}^{k} q_{l'}^{e_j} = 0] \tag{6.14}$$

We can easily prove that $\mathbb{P}[\sum_{j=1}^{k} q_r^{e_j} = 1] = \frac{1}{2}$. To prove this, we can use induction on k. This equation is valid for the base case $k = 1$. Assume that it is valid for $k - 1$. We have

$$\begin{aligned}\mathbb{P}[\sum_{j=1}^{k} q_r^{e_j} = 1] = \mathbb{P}[q_r^{e_k} = 1]\mathbb{P}[\sum_{j=1}^{k-1} q_r^{e_j} = 0] \\ + \mathbb{P}[q_r^{e_k} = 0]\mathbb{P}[\sum_{j=1}^{k-1} q_r^{e_j} = 1] = \frac{1}{2}\end{aligned}$$

Similarly, it is easy to prove that $\mathbb{P}[\sum_{j=1}^{k} q_{l'}^{e_j} = 0] = \frac{1}{2}$. Hence, equation (6.14) can be simplified to $\mathbb{P}[B_{\mathbf{x}}^r] = \mathbb{P}[\mathbf{Q}^\epsilon\mathbf{x} = \mathbf{e}_r] = 2^{-m}$. □

Lemma 6.5.10. The following inequalities hold for $1 \leq j \leq i$,

$$\frac{1}{i+1} 2^{iH(\frac{j}{i})} \leq \binom{i}{j} \leq 2^{iH(\frac{j}{i})} \tag{6.15}$$

where $H(\alpha)$ denotes the binary entropy function, i.e. $H(\alpha) = -\alpha \log_2(\alpha) - (1 - \alpha) \log_2(1 - \alpha)$.

Proof. The proof can be found in the appendix of [70]. □

We are now ready to prove the following theorem which shows that accessing a significant number of random queries in $\mathcal{Q}^\epsilon$ cannot help in reconstructing any of the base vectors for large m.

Theorem 6.5.11. Consider the set $\mathcal{Q}^\epsilon = \{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_{A^\epsilon}\}$ of $A^\epsilon = m + \lceil \log_2(\frac{1}{\epsilon}) \rceil$ statistically independent random uniform query vectors. For large enough values of m with probability arbitrarily close to 1, none of the base vectors exist in the span of any subset $\mathcal{Q}_l \subset \mathcal{Q}^\epsilon$ with cardinality of at most $l = \lfloor \delta m \rfloor$ where $\delta < 0.5$.

Proof. Consider any base vector $\mathbf{e}_r$ and a non-zero vector $\mathbf{x} \in \mathbb{F}_2^{A^\epsilon}$. For this vector, computing $\mathbf{Q}^\epsilon \mathbf{x}$ in $\mathbb{F}_2$ is equivalent to adding a subset of columns of $\mathbf{Q}^\epsilon$ whose set of indices is equal to the set of indices of non-zero elements in $\mathbf{x}$. If $\mathbf{Q}^\epsilon \mathbf{x} = \mathbf{e}_r$ for some $\mathbf{x} \in \mathbb{F}_2^{A^\epsilon}$, then any subset $\mathcal{Q}_l \subset \mathcal{Q}^\epsilon$ which contains the column vectors of $\mathbf{Q}^\epsilon$ whose set of indices is equal to the set of indices of non-zero elements in $\mathbf{x}$ also spans $\mathbf{e}_r$. In fact, the number of non-zero elements of $\mathbf{x}$ or Hamming weight of $\mathbf{x}$ (i.e., $\mathrm{Ham}(\mathbf{x})$) is equal to the number of vectors that should be added to reconstruct $\mathbf{e}_r$.

Consider all vectors $\mathbf{x} \in \mathbb{F}_2^{A^\epsilon}$ with Hamming weight less than or equal to $l = \lfloor \delta m \rfloor$ where $\delta < 0.5$. Lemma 6.5.9 shows that for any $\mathbf{x}$, we have $\mathbb{P}[B_{\mathbf{x}}^r] = 2^{-m}$. Therefore, the asymptotic probability of existence of a subset $\mathcal{Q}_l \subset \mathcal{Q}^\epsilon$ with a cardinality of at most $l = \lfloor \delta m \rfloor$ which spans $\mathbf{e}_r$ for large values of m can be found as

$$\lim_{m\to\infty} \mathbb{P}[\exists \mathcal{Q}_l \subseteq \mathcal{Q}^\epsilon | \mathrm{card}\{\mathcal{Q}_l\} \leq l = \lfloor \delta m \rfloor, \mathbf{e}_r \in \mathrm{span}\{\mathcal{Q}_l\}],$$
$$= \lim_{m\to\infty} \mathbb{P}[\bigcup_{\mathbf{x}\in\mathbb{F}_2^{A^\epsilon}, \mathrm{Ham}(\mathbf{x})\leq l} B_{\mathbf{x}}^r] \overset{(a)}{\leq} \lim_{m\to\infty} \sum_{\mathbf{x}\in\mathbb{F}_2^{A^\epsilon}, \mathrm{Ham}(\mathbf{x})\leq l} \mathbb{P}[B_{\mathbf{x}}^r],$$

$$
\begin{aligned}
&\overset{(b)}{=} \lim_{m\to\infty} \sum_{i=1}^{l} \binom{A^\epsilon}{i} 2^{-m} \overset{(c)}{\leq} \lim_{m\to\infty} l \binom{A^\epsilon}{l} 2^{-m}, \\
&\overset{(d)}{\leq} \lim_{m\to\infty} l \binom{m}{l} 2^{-m} = \lim_{m\to\infty} \lfloor \delta m \rfloor \binom{m}{\lfloor \delta m \rfloor} 2^{-m}, \\
&\overset{(e)}{\leq} \lim_{m\to\infty} \delta m 2^{mH\left(\frac{\lfloor \delta m \rfloor}{m}\right)} 2^{-m} \overset{(f)}{\leq} \lim_{m\to\infty} \delta m 2^{-m(1-H(\delta))} \overset{(g)}{=} 0,
\end{aligned}
$$

where inequality (a) comes from the *union bound* and (b) holds by using Lemma 6.5.9 and counting all the vectors $\mathbf{x}$ with Hamming weight less than $l = \lfloor \delta m \rfloor$ and inequality (e) comes from Lemma 6.5.10. Notice that (c), (d), (e) and (f) are only valid for cases when $\delta < 0.5$. This shows that the probability of existence of any desired base in the span of any subset of vectors with cardinality less than $\lfloor \delta m \rfloor$ goes to zero as m grows if $\delta < 0.5$. □

In the following theorem we will use the result of Theorem 6.5.11 to prove that accessing a large number of queries cannot reveal any information about the requested content index.

Theorem 6.5.12. For every subset $\mathcal{Q}_l \subset \mathcal{Q}^\epsilon$ with cardinality at most $l = \lfloor \delta m \rfloor$ where $\delta < 0.5$ we have

$$
\lim_{m\to\infty} I(R\ ;\ \mathcal{Q}_l) = 0. \tag{6.16}
$$

Proof. Let D_l be the event that none of the base vectors exist in the span of any subset $\mathcal{Q}_l \subset \mathcal{Q}^\epsilon$ with cardinality at most l. Let $\mathbf{e}_R$ be the equivalent random base vector which is uniquely defined by the requested content index R. If D_l happens, then for every

subset $\mathcal{Q}_l \subset \mathcal{Q}^\epsilon$ there should exist $k_l \geq 1$ random vectors $\mathbf{q}_{o_1}, \mathbf{q}_{o_2}, \ldots, \mathbf{q}_{o_{k_l}} \in \mathcal{Q}^\epsilon - \mathcal{Q}_l$ such that $\mathbf{e}_R \notin \text{span}\{\mathcal{Q}_l\}$ but $\mathbf{e}_R \in \text{span}\{\mathcal{Q}_l \cup \{\mathbf{q}_{o_i}, \mathbf{q}_{o_2}, \ldots, \mathbf{q}_{o_{k_l}}\}\}$. Hence, for any $\mathcal{Q}_l \subset \mathcal{Q}^\epsilon$, any representation of $\mathbf{e}_R$ in terms of random queries should have a form of

$$\mathbf{e}_R = \sum_{\mathbf{q}_j \in \mathcal{Q}_l} \theta_j \mathbf{q}_j + \sum_{i=1}^{k_l} \mathbf{q}_{o_i} \tag{6.17}$$

Where θ_i's are equal to zero or one. Hence, for every $\mathcal{Q}_l \subset \mathcal{Q}^\epsilon$ we should have

$$\begin{aligned} I(R; \mathcal{Q}_l | D_l) &= H(R|D_l) - H(R|\mathcal{Q}_l, D_l) \\ &\overset{(a)}{=} H(R|D_l) - H(R|D_l) = 0 \end{aligned} \tag{6.18}$$

where (a) comes from the independence of the content index (or its equivalent base vector $\mathbf{e}_R$) and the random queries. Using the chain rule for mutual information, we arrive at

$$\begin{aligned} I(R; \mathcal{Q}_l, D_l) &= I(R; D_l) + I(R; \mathcal{Q}_l | D_l) \\ &= I(R; \mathcal{Q}_l) + I(R; D_l | \mathcal{Q}_l). \end{aligned} \tag{6.19}$$

Combining equations (6.18) and (6.19) results in

$$\begin{aligned} I(R; \mathcal{Q}_l) &= I(R; D_l) - I(R; D_l | \mathcal{Q}_l) \\ &= H(D_l) - H(D_l|R)) - H(D_l|\mathcal{Q}_l) + H(D_l|R, \mathcal{Q}_l) \\ &\leq H(D_l) + H(D_l|R, \mathcal{Q}_l) \leq 2H(D_l). \end{aligned} \tag{6.20}$$

Theorem 6.5.11 shows that when $m \to \infty$ with probability close to one none of the base vectors exist in the span of any subset $\mathcal{Q}_l \subset \mathcal{Q}^\epsilon$ of cardinality less than or equal to $l = \lfloor \delta m \rfloor$ for $\delta < 0.5$. Hence, this theorem shows that event D_l will happen with

probability one when $m \to \infty$. Therefore, $\lim_{m\to\infty} H(D_l) = 0$ and thus for any subset $\mathcal{Q}_l \subset \mathcal{Q}^\epsilon$ of cardinality less than or equal to $l = \lfloor \delta m \rfloor$ for $\delta < 0.5$ we have

$$\lim_{m\to\infty} I(R; \mathcal{Q}_l) \leq 2 \lim_{m\to\infty} H(D_l) = 0 \tag{6.21}$$

This proves the theorem. □

Remark 6.5.13. In practice the user generates enough number of random vectors to span the whole m-dimensional space. Hence, it has a set $\mathcal{Q} = \{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_A\}$ of $A \geq m$ total random vectors. Then it chooses a subset $\mathcal{Q}^{\text{full}} = \{\mathbf{q}_{t_1}, \mathbf{q}_{t_2}, \ldots, \mathbf{q}_{t_m}\} \subseteq \mathcal{Q}$ of m linearly independent vectors from them and use them as its query vectors. This way it is guaranteed that the m queries will span the whole space of $\mathbb{F}_2^m$ and any base vector $\mathbf{e}_r$ can be represented in terms of these independent query vectors as

$$\mathbf{e}_r = \sum_{k=1}^{m} d_k \mathbf{q}_{t_k} \tag{6.22}$$

The following lemma shows that the average required number of random vectors to span $\mathbb{F}_2^m$ is very close to m so in practice only a few number of random queries more than m is needed to span $\mathbb{F}_2^m$.

Lemma 6.5.14. If $\mathbf{q}_j$ is a random vector belonging to $\mathbb{F}_2^m$ with elements having uniform distribution, the average minimum number of vectors $\mathbf{q}_j$ to span the whole space of $\mathbb{F}_2^m$ equals

$$\mathbb{E}_q = m + \sum_{i=1}^{m} \frac{1}{2^i - 1} = m + \gamma, \tag{6.23}$$

where γ asymptotically approaches the Erdős–Borwein constant (≈ 1.6067).

Proof. The proof can be found in [57]. □

Remark 6.5.15. Since $\mathcal{Q}^{\text{full}} \subseteq \mathcal{Q}$, if any vector $\mathbf{e}_r$ does not exist in the span of any subset $\mathcal{Q}_l \subset \mathcal{Q}$, of l random query vectors, it will not exist in the span of any subset $\mathcal{Q}_l \subset \mathcal{Q}^{\text{full}}$ of l random query vectors in $\mathcal{Q}^{\text{full}}$ too. So, Theorems 6.5.11 and 6.5.12 remain valid for this choice of random queries too. This means that in practice, every base vector is guaranteed to exist in the span of the m query vectors but none of the base vectors exist in the span of any subset $\mathcal{Q}_l \subset \mathcal{Q}$ with probability close to one if $l < \lfloor \delta m \rfloor$ for $\delta < 0.5$.

6.5.2 Responding to Queries

In this section, we assume that the user has chosen m linearly independent random query vectors in $\mathcal{Q}^{\text{full}}$ and wants to download the rth content. Since $\mathcal{Q}^{\text{full}}$ is a set of vectors which spans the whole space of $\mathbb{F}_2^m$, the user can expand the base vector $\mathbf{e}_r$ in terms of the query vectors in $\mathcal{Q}^{\text{full}}$ as mentioned in (6.22). Hence, the requested content can be expanded in terms of query vectors as

$$f_r = \mathbf{f}\mathbf{e}_r = \mathbf{f}\left(\sum_{k=1}^{m} d_k \mathbf{q}_{t_k}\right) = \sum_{k=1}^{m} d_k \mathbf{f}\mathbf{q}_{t_k} \tag{6.24}$$

where $d_k \in \mathbb{F}_2$ is either 0 or 1. Based on equation (6.24) the user requests some parts of the desired content from each RG so that none of the RGs can understand any information about the requested content.

To accomplish PIR, the user partitions the set of random queries $\mathbf{q}_{t_k}$ whose corresponding decoding gains d_k are non-zero into a disjoint subsets $\mathcal{Q}_1, \mathcal{Q}_2, \ldots, \mathcal{Q}_a$. The choice of number of subsets (i.e. a) depends on the number of colluding servers.

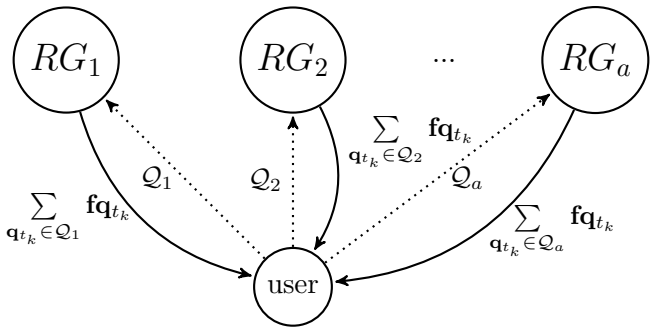

Figure 6.1: Multiple RGs respond to queries sent from the user. This allows the user to privately download its desired content while a significant number of colluding servers can achieve no information about the downloaded content.

Each subset of queries is then sent to a different RG as depicted in Figure 6.1. Therefore, the requested content can be retrieved as

$$f_r = \sum_{\substack{\mathbf{q}_{t_k} \in \mathcal{Q}^{\text{full}} \\ d_k \neq 0}} \mathbf{fq}_{t_k} = \sum_{i=1}^{a} \sum_{\mathbf{q}_{t_k} \in \mathcal{Q}_i} \mathbf{fq}_{t_k} \tag{6.25}$$

The ultimate goal in PIR is to prevent any colluding group of servers to gain information about the requested content index. Assume that the number of colluding servers is b. If any two colluding servers lie within the same RG, they receive the same subset of queries from the user. Therefore, without loss of generality we consider the worst scenario in which all the colluding servers lie within different RGs and all these b colluding servers are able to collaboratively obtain all the queries $\mathcal{Q}_1, \mathcal{Q}_2, \ldots, \mathcal{Q}_b$. Based on Theorems 6.5.11 and 6.5.12, if the number of all query vectors in $\mathcal{Q}_l = \cup_{i=1}^{b} \mathcal{Q}_i$ is less than $\lfloor \delta m \rfloor$ for some $\delta < 0.5$, then no information can be achieved about the requested content index. This provides significant PIR capability for this technique.

Notice that since RGs have full rank encoding matrices, they can respond to any query that they receive. Assume that RG $\mathcal{J}_i$ with the full rank encoding matrix

$\mathbf{V} = [\mathbf{V}_1\ \mathbf{V}_2\ \ldots\ \mathbf{V}_{J_i}]$ receives the set of queries $\mathcal{Q}_i$. This RG needs to send $\sum_{\mathbf{q}_{t_k} \in \mathcal{Q}_i} \mathbf{f}\mathbf{q}_{t_k}$ to the user. It can solve the linear equation

$$\mathbf{V}\mathbf{p}_i = \sum_{\mathbf{q}_{t_k} \in \mathcal{Q}_i} \mathbf{q}_{t_k} \tag{6.26}$$

in Galois Field $\mathbb{F}_2$ for $\mathbf{p}_i$ as

$$\mathbf{p}_i = \mathbf{V}^T \left(\mathbf{V}\mathbf{V}^T\right)^{-1} \left(\sum_{\mathbf{q}_{t_k} \in \mathcal{Q}_i} \mathbf{q}_{t_k} \right). \tag{6.27}$$

Similar to before the server $\mathcal{N}_s$ in the RG $\mathcal{J}_i$ which has already acquired all the information in matrix $\mathbf{V}$, computes the overal query decoding solution $\mathbf{p}_i$ which is a vector of size $J_i h \times 1$. If this vector is divided into J_i equal size pieces as $\mathbf{p}_i = [\mathbf{p}_i^1\ \mathbf{p}_i^2\ \ldots\ \mathbf{p}_i^{J_i}]^T$, then the server $\mathcal{N}_s$ sends the jth portion of $\mathbf{p}_i$ to server $\mathcal{N}_j$ in the RG $\mathcal{J}_i$. More precisely, server $\mathcal{N}_j$ receives a query response vector $\mathbf{p}_i^j$ of size $h \times 1$ from $\mathcal{N}_s$ for each $j = 1, 2, \ldots, J_i$. Then the server $\mathcal{N}_j$ sends $\mathbf{f}\mathbf{V}_j\mathbf{p}_i^j$ back to the coordinating server $\mathcal{N}_s$. The coordinating server $\mathcal{N}_s$ then aggregates all the data received from multiple servers in the RG to construct $\sum_{\mathbf{q}_{t_k} \in \mathcal{Q}_i} \mathbf{f}\mathbf{q}_{t_k}$ as

$$\sum_{\mathbf{q}_{t_k} \in \mathcal{Q}_i} \mathbf{f}\mathbf{q}_{t_k} = \mathbf{f}\mathbf{V}\mathbf{p}_i = \sum_{j=1}^{J_i} \mathbf{f}\mathbf{V}_j\mathbf{p}_i^j. \tag{6.28}$$

The coordinating server $\mathcal{N}_s$ in the RG $\mathcal{J}_i$ then transmits $\sum_{\mathbf{q}_{t_k} \in \mathcal{Q}_i} \mathbf{f}\mathbf{q}_{t_k}$ to the user.

Each RG only transmits one encoded file to the user. However all the servers within an RG need to collaborate with each other prior to responding to the queries sent from the user. Notice that communication between the servers are carried using high bandwidth fiber optic links while transmissions from the servers to the user are performed over low bandwidth links. In our computation of communication cost for

achieving PIR, we only consider communication between the servers and the user in the low bandwidth links.

Remark 6.5.16. It is worth mentioning that In our work the coordinations between servers in an RG is necessary because the servers do not have full storage capacity to store all the contents. In fact if we also assume each server has high storage capacity similar to [92], then each server can act as an RG and there will be no communications between servers.

6.5.3 Trade-off Between Communication Cost and Privacy Level

In order to achieve PIR, each user needs to download more information. This additional bandwidth utilization is referred to as *communication Price of Privacy (cPoP)* [92] which is defined as follows. Note that in this work, the cost of sending queries are ignored because it is assumed that the size of contents are significantly higher than the size of the queries.

Definition 6.5.17. The communication Price of Privacy (cPoP) is the ratio of the total number of bits downloaded by the user from the servers to the size of the requested file.

To explain the trade-off between communication cost and level of privacy, assume that the user divides the queries into a equal size groups of queries and sends each group of queries to a different RG. Each RG should respond to at most $\lceil \frac{m}{a} \rceil$ queries. If b RGs collude to gain some information about the requested content index, then they

will have access to a total of at most $b\lceil \frac{m}{a} \rceil$ queries. We proved in Theorem 6.5.12 that knowing $\lfloor \delta m \rfloor$ queries asymptotically gives no information about the requested content index if $\delta < 0.5$. Hence, if $b < \frac{a}{2}$, then the colluding RGs will get no information about the requested content index. Therefore, if less than half of the RGs collude to gain some information about the requested content index, they cannot gain any information. We can increase a to get the maximum possible level of privacy. However, the downside of increasing a is that the communication Price of Privacy (cPoP) will also increase.

As discussed earlier, if the queries are sent to a RGs then a responses from these RGs are required to retrieve a content. Since each RG transmits an encoded file of size M bits to the user the total number of bits downloaded by the user will be equal to aM and therefore the cPoP will be equal to $aM/M = a$. Figure 6.2 shows that as a increases, both the PIR strength and the cPoP increase linearly.

6.5.4 Full Size Servers

Assume that the servers have large storage capability such that each RG is only composed of 1 server. Our assumption of full rank encoding scheme guarantees that servers with storage ability of $h \geq m$ encoded files can be used to retrieve any desired content. In [92], the authors studied the use of MDS codes for PIR. They considered full size storage systems with MDS codes and they considered the case when only one of the databases is compromised. They proposed a PIR technique in which a cPoP of $\frac{1}{1-R}$ can be achieved in full size databases where R is the MDS code rate. To compare our results with [92], notice that if we assume that there is only one malicious server

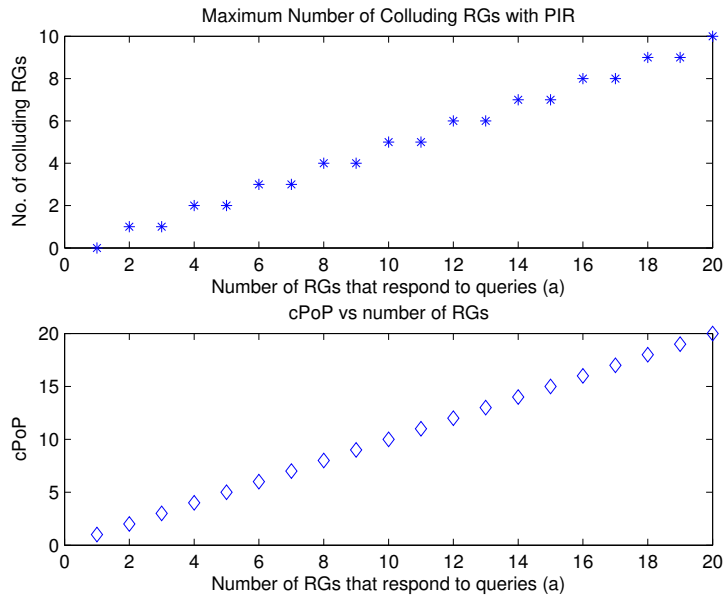

Figure 6.2: As the maximum number of colluding RGs increases, the average required communication Price of Privacy (cPoP) to maintain privacy increases.

in the cloud, then we can choose any two servers and send half of the queries to each one of them. This way we have a cPoP of 2 which is better than the results in [92] for $R > 1/2$.

6.6 Simulation

To numerically verify the results proved in section 6.5, we created m linearly independent random query vectors which are used to expand the bases. Figure 6.3 demonstrates the probability of the event that at least one of the base vectors exists in the span of $l = \lfloor \delta m \rfloor$ vectors for $\delta = 0.1, 0.2, 0.3$ and 0.4. Consistent with our results in section 6.5, the probability of the event that a base exists in the span of any set of

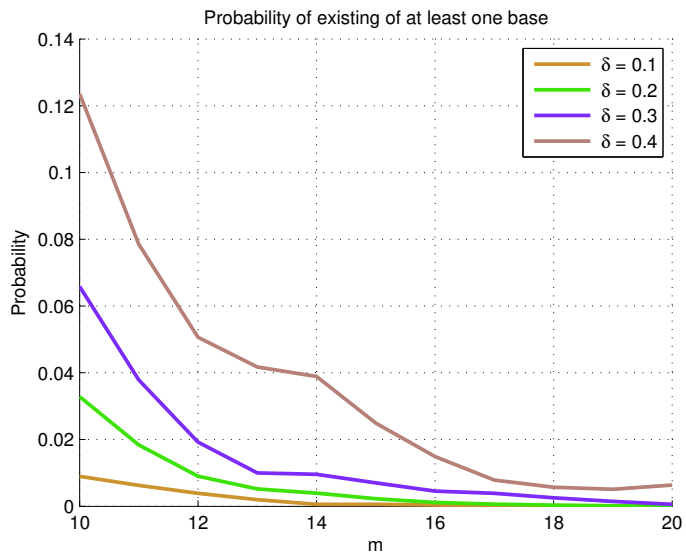

Figure 6.3: Probability of the event that at least one base exists in the span of any subset of $l = \lfloor \delta m \rfloor$ random vectors.

$l = \lfloor \delta m \rfloor$ vectors goes quickly to zero.

It is proved [98] that the problem of finding the minimum spanning set of vectors is NP-Complete. It is even proved [30] that this problem is NP-Hard to approximate. Therefore, in general it is NP-Hard to find out if a given base exists in the span of at most $l = \lfloor \delta m \rfloor$ vectors out of the m vectors. For our simulations we have used a brute force approach to check if a given base exists in the span of at most $l = \lfloor \delta m \rfloor$ vectors out of the m existing random query vectors where $m \leq 20$.

Chapter 7

Conclusions

In chapter 2, we introduced Modified Index Coding (MIC). MIC is a more general concept than original index coding problem which can be applied to both wireless multi-hop communications and wired networks. We applied MIC to information centric networks (ICN). The result was a hybrid caching scheme in ICN that consists of a caching at the routers and distributed caching at the nodes. The purpose of caching in routers is similar to the original concept of caching in ICN but we transmit encoded messages instead of the data itself. The purpose of distributed caching is to have replicas of the popular messages so that when we transmit new requested messages, we can combine multiple messages in order to serve multiple users with a single transmission. We proved that this combination results in significant transmission reductions and capacity improvements.

In chapter 3 an efficient order optimal content delivery approach is proposed for future wireless cellular systems. We take advantage of femtocaches [37] and multihop

communication using index coding. We proved that using index coding is very efficient technique by taking advantage of side information stored in users. Further, it was shown that under Zipfian content distribution, linear index coding could be order optimal. Our simulation result demonstrates the gains that can be achieved with this approach. The proposed technique in [37] requires deployment of large number of femtocaches in order to cache the contents locally and retrieve them when needed. We propose to use a multihop transmission scheme which significantly reduces the femtocache deployment costs compared to [37]. Further, we show that significant capacity gains can be achieved through the use of index codes. The optimal index coding solution is an NP-Hard problem [64]. We propose a simple heuristic to perform the task of index coding and we will show that this heuristic which is based on graph coloring is asymptotically capable of achieving maximum index coding capacity.

In chapter 4 we studied the throughput capacity of cellular networks with femtocaches using decentralized uncoded and coded content cache placement. We proposed multihop communications to take away some of the communication burden from the helpers and base station and transfer it to users with storage capability. We proved that multihop communication together with clever use of cache placement strategy can increase the throughput capacity of these networks. We proposed a novel decentralized coded caching scheme based on Random Linear Fountain (RLF) codes. In this technique, each user independently caches a random combination of all the other files in a decentralized manner. Our proposed scheme improves the efficiency of femtocaches by taking advantage of multihop communications and index coding. The proposed decen-

tralized coded content cache placement scheme can increase the throughput capacity by a factor of $(\log n)^2$ over decentralized uncoded content caching. Using our proposed decentralized coded content cache placement scheme, we computed the throughput capacity of cellular networks operating under a Zipfian content request distribution.

In chapter 5 we introduced decentralized coded caching strategy in wireless ad hoc networks. The capacity of this approach is compared with that of uncoded caching for proactive and reactive routing protocols. While with proactive routing protocol uncoded caching outperforms coded caching, coded caching performs better with reactive routing protocol. Interestingly, it was shown that by choosing any random direction, close to optimum number of hops can be obtained to retrieve any content in coded caching. It is shown that when the number of cached files is very large this technique satisfies in the perfect secrecy conditions proposed by Shannon in [88] and therefore this technique is capable of achieving asymptotic perfect secrecy in wireless ad hoc networks. This provides an information theoretically secure solution for caching proprietary contents in wireless networks which is immune to attackers in time as opposed to computationally security which may be broken with time. We have also studied the cache hit problem and shown that the cache hit probability for any desired content will be significantly higher in the proposed technique compared to uncoded caching. Therefore, this technique is able to reduce the problem of overcaching in the networks significantly. An efficient and secure cache update algorithm is also proposed. The theoretical results are validated with simulations. The results in chapter 5 are achieved with minimal overhead in contrast to works like [44, 45] where higher overhead is required to find the

content and route.

In chapter 6, we studied the problems of security and private information retrieval in distributed storage systems which are using a full rank encoding scheme based on Random Linear Fountain (RLF) codes. We have proposed an approach based on uniform random queries to achieve information theoretic PIR property. We have proved that our proposed technique can asymptotically achieve perfect secrecy for a distributed storage system. Our proposed solution is robust against a significant number of colluding servers in the network. We have also shown that our technique can outperform MDS codes for storage systems in terms of PIR cost for certain regimes.

Bibliography

[1] expressive internet architecture project. *http://www.cs.cmu.edu/ xia/*.

[2] Nebula project. *http://nebula.cis.upenn.edu*.

[3] `http://spectrum.ieee.org/telecom/wireless/millimeter-waves-may-be-the-future-of-5g-`

[4] Amendments in IEEE 802.11ad™ enable multi-gigabit data throughput and groundbreaking improvements in capacity. `https://standards.ieee.org/news/2013/802.11ad.html`, 2013. [Online; accessed 9-October-2015].

[5] Named data networking. *http://named-data.net/*, August 2010.

[6] Psirp: publish-subscribe internet routing paradigm. *http://psirp.org/*, Jan. 2008.

[7] Ziv Bar-Yossef, Yitzhak Birk, TS Jayram, and Tomer Kol. Index coding with side information. *Information Theory, IEEE Transactions on*, 57(3):1479–1494, 2011.

[8] Doug Beaver, Sanjeev Kumar, Harry C Li, Jason Sobel, Peter Vajgel, et al. Finding a needle in haystack: Facebook's photo storage. In *OSDI*, volume 10, pages 1–8, 2010.

[9] Yitzhak Birk and Tomer Kol. Informed-source coding-on-demand (iscod) over broadcast channels. In *INFOCOM'98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1257–1264. IEEE, 1998.

[10] Yitzhak Birk and Tomer Kol. Coding on demand by an informed source (iscod) for efficient broadcast of different supplemental data to caching clients. *IEEE/ACM Transactions on Networking (TON)*, 14(SI):2825–2830, 2006.

[11] Matthieu Bloch and Joao Barros. *Physical-layer security: from information theory to security engineering.* Cambridge University Press, 2011.

[12] Federico Boccardi, Robert W Heath, Aurelie Lozano, Thomas L Marzetta, and Petar Popovski. Five disruptive technology directions for 5G. *Communications Magazine, IEEE*, 52(2):74–80, 2014.

[13] Béla Bollobás. The chromatic number of random graphs. *Combinatorica*, 8(1):49–55, 1988.

[14] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. On the implications of Zipf's law for web caching. Technical report, Citeseer, 1998.

[15] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web caching and Zipf-like distributions: Evidence and implications. In *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 126–134, New York, NY, 1999. IEEE.

[16] Ning Cai and Raymond W Yeung. Secure network coding. In *Proceedings of IEEE International Symposium on Information Theory, ISIT*, page 323. IEEE, 2002.

[17] Ning Cao, Shucheng Yu, Zhenyu Yang, Wenjing Lou, and Y. Thomas Hou. LT codes-based secure and reliable cloud storage service. In *Proceedings of the IEEE INFOCOM 2012, Orlando, FL, USA, March 25-30*, pages 693–701, 2012.

[18] M. Cha, H. Kwak, P. Rodriguez, Y. Ahn, and S. Moon. I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement.*, 2007.

[19] Terence H. Chan, Siu-Wai Ho, and Hirosuke Yamamoto. Private information retrieval for coded storage. In *IEEE International Symposium on Information Theory, ISIT 2015, Hong Kong, China, June 14-19, 2015*, pages 2842–2846, 2015.

[20] Vikram Chandrasekhar, Jeffrey G Andrews, and Alan Gatherer. Femtocell networks: a survey. *Communications Magazine, IEEE*, 46(9):59–67, 2008.

[21] Mohammad Asad R Chaudhry, Zakia Asad, Alex Sprintson, and Michael Langberg. On the complementary index coding problem. In *Proceedings of Information Theory (ISIT), IEEE International Symposium on*, pages 244–248, Saint Petersburg, Russia, 2011. IEEE.

[22] Mohammad Asad R Chaudhry and Alex Sprintson. Efficient algorithms for index coding. In *INFOCOM Workshops, IEEE*, pages 1–4, Phoenix, AZ, 2008. IEEE.

[23] Zhi Chen. Fundamental limits of caching: Improved bounds for small buffer users. *arXiv preprint arXiv:1407.1935*, 2014.

[24] Fan Cheng and Vincent YF Tan. A numerical study on the wiretap network with a simple network topology. *IEEE Transactions on Information Theory*, 62(5):2481–2492, 2016.

[25] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, 23-25 October 1995*, pages 41–50, 1995.

[26] Theodoros K. Dikaliotis, Alexandros G. Dimakis, and Tracey Ho. Security in distributed storage systems by communicating a logarithmic number of bits. In *IEEE International Symposium on Information Theory, ISIT 2010, June 13-18, 2010, Austin, Texas, USA, Proceedings*, pages 1948–1952, 2010.

[27] Alexandros G. Dimakis, Brighten Godfrey, Yunnan Wu, Martin J. Wainwright, and Kannan Ramchandran. Network coding for distributed storage systems. *IEEE Trans. Information Theory*, 56(9):4539–4551, 2010.

[28] Alexandros G. Dimakis, Vinod M. Prabhakaran, and Kannan Ramchandran. Distributed fountain codes for networked storage. In *Proceedings of International Conference on Acoustics Speech and Signal Processing, ICASSP 2006, Toulouse, France, May 14-19*, pages 1149–1152, 2006.

[29] Alexandros G. Dimakis, Kannan Ramchandran, Yunnan Wu, and Changho Suh.

A survey on network codes for distributed storage. *Proceedings of the IEEE*, 99(3):476–489, 2011.

[30] Ilya Dumer, Daniele Micciancio, and Madhu Sudan. Hardness of approximating the minimum distance of a linear code. *IEEE Trans. Information Theory*, 49(1):22–37, 2003.

[31] Michelle Effros, Salim El Rouayheb, and Michael Langberg. An equivalence between network coding and index coding. *Information Theory, IEEE Transactions on*, 61(5):2478–2487, 2015.

[32] Salim El Rouayheb, Alex Sprintson, and Costas Georghiades. On the index coding problem and its relation to network coding and matroid theory. *Information Theory, IEEE Transactions on*, 56(7):3187–3195, 2010.

[33] P Erdős and L Pósa. On the maximal number of disjoint circuits of a graph. *Publ. Math. Debrecen*, 9:3–12, 1962.

[34] Paul Erdős. On a classical problem of probability theory. 1961.

[35] Seyed Kaveh Fayazbakhsh, Yin Lin, Amin Tootoonchian, Ali Ghodsi, Teemu Koponen, Bruce Maggs, KC Ng, Vyas Sekar, and Scott Shenker. Less pain, most of the gain: Incrementally deployable ICN. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 147–158. ACM, 2013.

[36] Michael R Garey and David S Johnson. *Computers and intractability*. W. H. Freeman, San Francisco, 2002.

[37] Negin Golrezaei, Karthikeyan Shanmugam, Alexandros G Dimakis, Andreas F Molisch, and Giuseppe Caire. Femtocaching: Wireless video content delivery through distributed caching helpers. In *INFOCOM, Proceedings IEEE*, pages 1107–1115, Orlando, FL, 2012. IEEE.

[38] Jad Hachem, Nikhil Karamchandani, and Suhas Diggavi. Multi-level coded caching. In *Information Theory (ISIT), 2014 IEEE International Symposium on*, pages 56–60. IEEE, 2014.

[39] Sajad Hataminia, Saeed Vahidian, Mohammadali Mohammadi, and Mahmoud Ahmadian-Attari. Performance analysis of two-way decode-and-forward relaying in the presence of co-channel interferences. *IET Commun.*, 8(18):3349–3356, Dec. 2014.

[40] Sajad Hatamnia, Saeed Vahidian, Sonia Aïssa, Benoit Champagne, and Mahmoud Ahmadian-Attari. Network-coded two-way relaying in spectrum-sharing systems with quality-of-service requirements. *IEEE Transactions on Vehicular Technology*, 66(2):1299–1312, 2017.

[41] Ishay Haviv and Michael Langberg. On linear index coding for random graphs. In *Information Theory Proceedings (ISIT), IEEE International Symposium on*, pages 2231–2235, Cambridge, MA, 2012. IEEE.

[42] Tracey Ho, Muriel Médard, Ralf Koetter, David R Karger, Michelle Effros, Jun

Shi, and Ben Leong. A random linear network coding approach to multicast. *Information Theory, IEEE Transactions on*, 52(10):4413–4430, 2006.

[43] Yan Hui. Method and apparatus for data scrambling/descrambling, May 29 2003. US Patent App. 09/997,639.

[44] Sang-Woon Jeon, Song-Nam Hong, Mingyue Ji, and Giuseppe Caire. Caching in wireless multihop device-to-device networks. In *Communications (ICC), 2015 IEEE International Conference on*, pages 6732–6737. IEEE, 2015.

[45] Sang-Woon Jeon, Song-Nam Hong, Mingyue Ji, Giuseppe Caire, and Andreas F Molisch. Wireless multihop device-to-device caching networks. *arXiv preprint arXiv:1511.02574*, 2015.

[46] Mingyue Ji, Giuseppe Caire, and Andreas F Molisch. Wireless device-to-device caching networks: Basic principles and system performance. *arXiv preprint arXiv:1305.5216*, 2013.

[47] Mingyue Ji, Giuseppe Caire, and Andreas F Molisch. Fundamental limits of caching in wireless D2D networks. *Information Theory, IEEE Transactions on*, 62(2):849–869, 2016.

[48] Norman Lloyd Johnson and Samuel Kotz. Urn models and their application; an approach to modern discrete probability theory. 1977.

[49] Nikhil Karamchandani, Urs Niesen, Mohammad Ali Maddah-Ali, and Suhas Dig-

gavi. Hierarchical coded caching. In *Information Theory (ISIT), IEEE International Symposium on*, pages 2142–2146, Honolulu, HI, 2014. IEEE.

[50] Mohsen Karimzadeh Kiskani, Bita Azimdoost, and Hamid Sadjadpour. Effect of social groups on the capacity of wireless networks. *Wireless Communications, IEEE Transactions on*, 15:3–13, 2016.

[51] Mohsen Karimzadeh Kiskani and Hamid Sadjadpour. Application of index coding in information-centric networks. In *Computing, Networking and Communications (ICNC), 2015 International Conference on*, pages 977–983. IEEE, 2015.

[52] Mohsen Karimzadeh Kiskani and Hamid R. Sadjadpour. Capacity of cellular networks with femtocache. In *Proceedings of the IEEE Conference on Computer Communications Workshops, INFOCOM Workshops, San Francisco, USA, April 10 - 15*, 2016.

[53] Mohsen Karimzadeh Kiskani and Hamid R Sadjadpour. Multihop caching-aided coded multicasting for the next generation of cellular networks. *IEEE Transactions on Vehicular Technology*, 66(3):2576–2585, 2017.

[54] Mohsen Karimzadeh Kiskani and Hamid R Sadjadpour. Secure and private cloud storage systems with random linear fountain codes. In *IEEE International Conference on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress*

(UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), San Francisco Bay Area, CA. IEEE, 2017.

[55] Mohsen Karimzadeh Kiskani and Hamid R Sadjadpour. A secure approach for caching contents in wireless ad hoc networks. *IEEE Transactions on Vehicular Technology*, 2017.

[56] Mohsen Karimzadeh Kiskani and Hamid R Sadjadpour. Secure coded caching in wireless ad hoc networks. In *Computing, Networking and Communications (ICNC), 2017 International Conference on*, pages 387–391. IEEE, 2017.

[57] Mohsen Karimzadeh Kiskani and Hamid R Sadjadpour. Throughput analysis of decentralized coded content caching in cellular networks. *IEEE Transactions on Wireless Communications*, 16(1):663–672, 2017.

[58] Mohsen Karimzadeh Kiskani, Zheng Wang, Hamid R Sadjadpour, Jose A Oviedo, and Jose Joaquin Garcia-Luna-Aceves. Opportunistic interference management: a new approach for multiantenna downlink cellular networks. *Wireless Communications and Mobile Computing*, 15(14):1837–1850, 2015.

[59] Donald E Knuth. Big omicron and big omega and big theta. *ACM Sigact News*, 8(2):18–24, 1976.

[60] Valentin Fedorovich Kolchin. *Random graphs*. Number 53. Cambridge University Press, 1999.

[61] Zhenning Kong, Salah A. Aly, and Emina Soljanin. Decentralized coding algorithms for distributed storage in wireless sensor networks. *IEEE Journal on Selected Areas in Communications*, 28(2):261–267, 2010.

[62] Sanjeev R Kulkarni and Pramod Viswanath. A deterministic approach to throughput scaling in wireless networks. *Information Theory, IEEE Transactions on*, 50(6):1041–1049, 2004.

[63] Siddhartha Kumar, Eirik Rosnes, and Alexandre Graell i Amat. Secure repairable fountain codes. *IEEE Communications Letters*, 20(8):1491–1494, 2016.

[64] Michael Langberg and Alex Sprintson. On the hardness of approximating the network coding capacity. *Information Theory, IEEE Transactions on*, 57(2):1008–1014, 2011.

[65] Namyoon Lee, Alexandros G Dimakis, and Robert W Heath. Index coding with coded side-information. *Communications Letters, IEEE*, 19(3):319–322, 2015.

[66] Derek Leong, Tracey Ho, and Rebecca Cathey. Optimal content delivery with network coding. In *Information Sciences and Systems, CISS, 43rd Annual Conference on*, pages 414–419, Baltimore, MD, 2009. IEEE.

[67] Luisa Lima, Muriel Médard, and Joao Barros. Random linear network coding: A free cipher? In *Proceedings of the International Symposium on Information Theory, ISIT 2007*, pages 546–550. IEEE, 2007.

[68] Jaime Llorca, Antonia M Tulino, Ke Guan, and Daniel Kilper. Network-coded

caching-aided multicast for efficient content delivery. In *Communications (ICC), IEEE International Conference on*, pages 3557–3562, Budapest, Hungary, 2013. IEEE.

[69] Michael Luby. LT codes. In *43rd Symposium on Foundations of Computer Science (FOCS 2002), 16-19 November 2002, Vancouver, BC, Canada, Proceedings*, page 271, 2002.

[70] David J. C. MacKay. Good error-correcting codes based on very sparse matrices. *IEEE Trans. Information Theory*, 45(2):399–431, 1999.

[71] David JC MacKay. Fountain codes. In *Communications, IEE Proceedings-*, volume 152, pages 1062–1068. IET, 2005.

[72] Mohammad Ali Maddah-Ali and Urs Niesen. Fundamental limits of caching. *Information Theory, IEEE Transactions on*, 60(5):2856–2867, 2014.

[73] Mohammad Ali Maddah-Ali and Urs Niesen. Decentralized coded caching attains order-optimal memory-rate tradeoff. *IEEE/ACM Transactions on Networking (TON)*, 23(4):1029–1040, 2015.

[74] Marie-Jose Montpetit, Cedric Westphal, and Dirk Trossen. Network coding meets information-centric networking: an architectural case for information dispersion through native network coding. In *Proceedings of the 1st ACM workshop on Emerging Name-Oriented Mobile Networking Design-Architecture, Algorithms, and Applications*, pages 31–36, Hilton Head, SC, 2012. ACM.

[75] Mohammadreza Mousaei and Besma Smida. Optimizing pilot overhead for ultra-reliable short-packet transmission. *arXiv preprint arXiv:1705.02753*, 2017.

[76] Vahid Naghshin and Mark C Reed. On capacity and association area characterization in small cell-based multi-tier networks. *IEEE Wireless Communications Letters*, 4(5):505–508, 2015.

[77] Vahid Naghshin, Mark C Reed, and Neda Aboutorab. Coverage analysis of packet multi-tier networks with asynchronous slots. *IEEE Transactions on Communications*, 65(1):200–215, 2017.

[78] Michael J Neely, Arash Saber Tehrani, and Zhen Zhang. Dynamic index coding for wireless broadcast networks. *Information Theory, IEEE Transactions on*, 59(11):7525–7540, 2013.

[79] Urs Niesen and Mohammad Ali Maddah-Ali. Coded caching for delay-sensitive content. In *Communications (ICC), IEEE International Conference on*, pages 5559–5564, London, 2015. IEEE.

[80] Ramtin Pedarsani, Mohammad Ali Maddah-Ali, and Urs Niesen. Online coded caching. In *Communications (ICC), IEEE International Conference on*, pages 1878–1883, Sydney, NSW, 2014. IEEE.

[81] Li Peng, Song Guo, Shui Yu, and Athanasios V. Vasilakos. Codepipe: An opportunistic feeding and routing protocol for reliable multicast with pipelined network coding. In *Proceedings of IEEE INFOCOM*, pages 100–108, 2012.

[82] Mathew D Penrose. The longest edge of the random minimal spanning tree. *The annals of applied probability*, pages 340–361, 1997.

[83] Wei Quan, Xu Changqiao, Athanasios Vasilakos, Jianfeng Guan, Hongke Zhang, and Luigi Alfredo Grieco. Tb 2 f: Tree-bitmap and bloom-filter for a scalable and efficient name lookup in content-centric networking. In *IFIP Networking*, 2014.

[84] Theodore S Rappaport, Shu Sun, Rimma Mayzus, Hang Zhao, Yaniv Azar, Kangping Wang, George N Wong, Jocelyn K Schulz, Mathew Samimi, and Felix Gutierrez. Millimeter wave mobile communications for 5G cellular: It will work! *Access, IEEE*, 1:335–349, 2013.

[85] Hamid R. Sadjadpour. A new design for information centric networks. In *IEEE 48th Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6, 2014.

[86] Nihar B. Shah, K. V. Rashmi, and Kannan Ramchandran. One extra bit of download ensures perfectly private information retrieval. In *2014 IEEE International Symposium on Information Theory, Honolulu, HI, USA, June 29 - July 4, 2014*, pages 856–860, 2014.

[87] Karthikeyan Shanmugam, Negin Golrezaei, Alexandros G Dimakis, Andreas F Molisch, and Giuseppe Caire. Femtocaching: Wireless content delivery through distributed caching helpers. *Information Theory, IEEE Transactions on*, 59(12):8402–8413, 2013.

[88] Claude E Shannon. Communication theory of secrecy systems*. *Bell system technical journal*, 28(4):656–715, 1949.

[89] Yi-Sheng Shiu, Shih Yu Chang, Hsiao-Chun Wu, Scott C.-H. Huang, and Hsiao-Hwa Chen. Physical layer security in wireless networks: a tutorial. *IEEE Wireless Commun.*, 18(2):66–74, 2011.

[90] Amin Shokrollahi. Raptor codes. *IEEE Trans. Information Theory*, 52(6):2551–2567, 2006.

[91] Wolfgang Stadje. The collector's problem with group drawings. *Advances in Applied Probability*, pages 866–882, 1990.

[92] Razan Tajeddine and Salim El Rouayheb. Private information retrieval from MDS coded data in distributed storage systems. *arXiv preprint arXiv:1602.01458*, 2016.

[93] Tuan Tran, Thinh Nguyen, Bella Bose, and Vinodh Gopal. A hybrid network coding technique for single-hop wireless networks. *Selected Areas in Communications, IEEE Journal on*, 27(5):685–698, 2009.

[94] Vutha Va, Takayuki Shimizu, Gaurav Bansal, Robert W Heath Jr, et al. Millimeter wave vehicular communications: A survey. *Foundations and Trends® in Networking*, 10(1):1–113, 2016.

[95] Saeed Vahidian, Sonia Aïssa, and Sajad Hatamnia. Relay selection for security-constrained cooperative communication in the presence of eavesdropper's over-

hearing and interference. *IEEE Wireless Communications Letters*, 4(6):577–580, 2015.

[96] Saeed Vahidian, Maryam Najafi, Marzieh Najafi, and Fawaz S Al-Qahtani. Power allocation and cooperative diversity in two-way non-regenerative cognitive radio networks. *arXiv preprint arXiv:1705.02242*, 2017.

[97] Saeed Vahidian, Ehsan Soleimani-Nasab, Sonia Aïssa, and Mahmoud Ahmadian-Attari. Bidirectional AF relaying with underlay spectrum sharing in cognitive radio networks. *IEEE Transactions on Vehicular Technology*, 66(3):2367–2381, 2017.

[98] Alexander Vardy. The intractability of computing the minimum distance of a code. *IEEE Trans. Information Theory*, 43(6):1757–1766, 1997.

[99] Yongge Wang. LT codes for efficient and reliable distributed storage systems revisited. *arXiv preprint arXiv:1207.5542*, 2012.

[100] Zheng Wang, Hamid R Sadjadpour, JJ Garcia-Luna-Aceves, and Shirish S Karande. Fundamental limits of information dissemination in wireless ad hoc networks-part I: single-packet reception. *Wireless Communications, IEEE Transactions on*, 8(12):5749–5754, 2009.

[101] Alec Wolman, M Voelker, Nitin Sharma, Neal Cardwell, Anna Karlin, and Henry M Levy. On the scale and performance of cooperative web proxy caching. *ACM SIGOPS Operating Systems Review*, 33(5):16–31, 1999.

[102] Qinghua Wu, Zhenyu Li, and Gaogang Xie. Codingcache: multipath-aware CCN cache with network coding. In *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-Centric Networking*, pages 41–42, Hong Kong, 2013. ACM.

[103] Feng Xue and Panganamala R. Kumar. *Scaling Laws for Ad Hoc Wireless Networks: An Information Theoretic Approach*, volume 1. Now Publishers Inc, 2006.

[104] R Yeung, SY Li, Ning Cai, and Zhen Zhang. Network coding theory (foundations and trends in communications and information theory). *New York: Now*, 2006.