Matching-based Cache Placement Decision for 5G network caching

By

Shadi Sadeghpour Kharkan
B.Sc., Islamic Azad University, Iran
2005

A thesis presented to
Ryerson University
in partial fulfillment of the requirements
for the degree of

Master of Applied Science
in the program of Computer Networks

Toronto, Ontario, Canada

Abstract

Matching-based Cache Placement Decision for 5G network caching
Shadi Sadeghpour Kharkan
M.A.Sc, Computer Networks, Ryerson University, 2018

In this thesis, we present a cache placement scheme to deal with backhaul link constraint in Small Cell Network for 5G wireless network. We formulated the cache placement problem as a graph matching problem and presented an optimal file-helper matching algorithm. We defined stability criterion for the matching and found that our matching solution is stable in the sense that every helper finds at least one file to cache given that no file exceed minimum cache size.

We achieved a unique placement of a file within a cluster of helpers to increase the number of files cached within a cluster. Further, our experimental evaluation demonstrates that our algorithm increases local and neighbor hit ratios as compared to a random placement, which in turn significantly decreases the traffic that goes over the backhaul bottleneck link.

Acknowledgements

It is pleasure to thanks to Professor Muhammad Jaseemuddin who made this thesis possible and many insightful conversations during the development of ideas. Without his continuous support, enthusiasm, and encouragement, this study would hardly have been completed.

I also express my sincere gratitude to Professor Ma for his unwavering support and collegiality over the years.

Table of Content

List of Figures

List of Tables

Chapter 1

Introduction

One of the most effective ways to achieve high data throughput in a wireless network is using small cells in the network (SCN). Due to explosive demand for delivery of various contents specifically stream videos on portable devices, the idea of bringing the contents closer to the users was raised [1]. This type of the network which is called Femtocell or Picocell, are impaired by the capacity of backhaul link, since the capacity should be in the same order of access wireless links to prevent bottleneck in the process of delivering huge traffic generated by mobile users.

There is a huge body of work on the heterogeneous network (HeNets) or small cell networks (SCNs) from various aspects of cell associations, energy efficiency, and mobility management to LTE/Wi-Fi interworking.

Most of the current research in SCNs approach the problems in a reactive way meaning the caching decision is made in response to user's requests for the content. This approach requires expensive high-speed backhaul link to cope with the peak traffic demands, which is not sustainable as the rate of requests continuously grows in the network. In another approach, the storage at base-stations, content-awareness even social networking that affects the network usage, should be considered in reducing the traffic on the backhaul link. In this approach, a proactive solution can be designed that requires predicting users' context information to intelligently store contents to offload the backhaul and bring a high quality of service (QoS) for users [2].

With the prediction of future data requests and storing of data in the cache, the probability of serving a huge amount of users' requests without further downloading through backhaul link increases. Predicting future users demand and estimating the popularity of the contents, content can be stored in the cache of nodes that reside at the edge of the network. Consequently, proactively caching users' content at Small Base Stations (SBSs) would ease the backhaul load and improve the QOS from the user's point of view.

For this solution to work, providing a good caching technique is crucial and the importance of this point has brought many studies to a different setting and different objectives to cache at the base stations. Here when the concept of hierarchical caching comes up where the information theoretic approach applies to deal with.

Collaboration among small cells can efficiently reduce the cost of operation in a cellular network and improve the performance that aims to deploy collaborative caching [4].

Collaborative caching is proposed for proactive caching as well as reactive caching through offline and online settings [4]. It is found that content caching in a multicellular system by collaborating with their small cells can minimize the total cost paid by content provider [4].

Caching helpers (femtocells, Wi-Fi) with high storage capacity can be placed small cell base stations [1]. These helpers comprise a high bandwidth communication and storage capacity with low rate backhaul link. The localized high bandwidth communication capabilities enable them for high-frequency reuse by caching the popular contents and serve the requests from mobile user terminals (UEs).

There are some studies that show the potential of caching at the UEs side that can help to bring the contents even closer [3], e.g., Device to Device caching or local cache offloading. These concepts try to deal with the placement problem of caching contents in a wireless system.

Since caching the most popular files at the Small Base Stations (SBSs) is known to effectively increase the data throughput and bring the contents closer to the users. FemtoCaching is introduced as a novel architecture to use the concept of caching in a very practical way to deal with backhaul capacity constraints by reusing local storage at the SBSs for caching [9].

However, the effectiveness of caching largely depends on the reusability of the content stored in the cache. The decision for the placement of content (files) in the cache is crucial for improving the reusability of the cached file. In this thesis, we propose a placement algorithm based on the well-known graph matching problem in order to increase the probability of finding the requested files already cached at the helpers.

The thesis is organized as follows.

First, we discuss background work on caching with focus on cache placement in chapter 2. Then in chapter 3, we present our placement algorithm based on graph matching. We discuss the user and helper preferences and develop their preference matrices, then we describe the matching game and its requirement, and finally we provide our algorithm and its pseudo-code. In chapter 4, we evaluate the performance of our algorithm as well as the simulation results. Finally, we present some concluding remarks and future work in chapter 5.

Chapter 2

Background

This chapter discusses the concepts and research that are relevant to the work reported in this thesis. Section 2.1 gives an overview of the basic ideas of caching placement in a wireless networks. In section 2.2 proactive caching in small cell network is discussed. Section 2.3 provides the schemes related to caching placement for D2D assisted in wireless networks. Finally, collaborative small cellular network is reviewed in the last section.

2.1 Cache Placement

The dramatic growth of data traffic in the wireless network is due to downloading of high demand files that are requested every day. Comparative analysis of small cell architecture and conventional macro-cell architecture with the estimated growth demand in future motivated researchers to look for new technologies to cope with such traffic. One of the most promising approaches is introduced to reduce the cell size and bring the content closer to the user [1]. Small cell network becomes viable for localized communication and when communication resources are reused. Femto-cell architecture [1] was proposed to handle users' demand while using short-range links to the nearest small base station. These femto base stations (helpers) with high storage capacity are connected to the backbone network through low rate backhaul links. The bottleneck bandwidth of backhaul links can be mitigated by storing popular files and serving the requests through the backhaul link only when none of the helpers could serve them. In [1], the placement of these contents is discussed to answer the key question of which files should be placed in which helpers. The optimal caching policy is relatively simple when a user can communicate with only one helper because in this case each helper should cache the most popular files. However, when a user has connections to multiple helpers, caching policy becomes more challenging and the task of assigning caches to helpers becomes crucial.

Reference [1] presents the formalization of the caching placement problem and the design of an architecture for increasing the throughput of wireless video delivery network to reduce the backhaul traffic load.

2.1.1 Distributed Cache Placement

The scheme in [1] considers that all video files are requested randomly and they are redundant. It deals with the problem of finding a way of placing the files among helpers to minimize the average delay experienced by users. Figure 2.1 illustrates the relationship between users and helpers when four users and two helpers are engaged in communication. It is obvious from the figure that U1 and U2 prefer helper1 and U4 prefers helper 2 to cache their popular files. For U3, both helpers H1 and H2 are equally preferable and they together provide almost double the space for caching U3 as compared to a single helper.

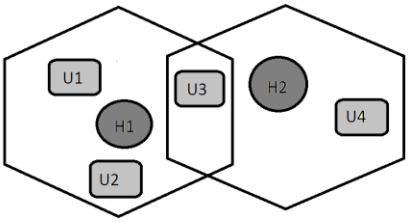

Figure 2. 1 Distributed caching and conflicting interests among users.

In general, for a system with *H* number of helpers, *K* user terminals and a library of *N* files, each user sends a request with the probability of p_n while relationship between users and helpers can be modeled as a bipartite graph. When a local helper receives a request, it first looks in its cache to find the requested file and serve it. If the local helper does not find the requested file in its cache, it sends the request to the BS that serves the request at higher delay.

In-network caching is proposed to increase the throughput of wireless contents (Videos) delivery and decrease the backhaul traffic load. Since the storage becomes cheap, caching at the base stations provides a low-cost solution of reducing the load in the backhaul links. The download cost can further be reduced by offloading the download traffic from the low-rate backhaul link at peak time through scheduling download requests at off-peak hours especially in case of proactive caching. In [1] the placement problem is formulated in terms of maximizing a monotone submodular function over matroid constraints and a greedy algorithm is introduced that solves the

problem in ½ of the optimal value. Researchers evaluated their scheme by simulating an LTE-based cellular network and using a real trace of YouTube requests. Their result shows almost 500% performance improvements making the proposed scheme a promising way of alleviating the bottlenecks in wireless video delivery.

2.2 Proactive Caching in Small Cell Networks

One of the key roles of employing small-cell base stations in the cellular wireless network is to take the advantage of using their capacity and valuable coverage to avoid the bottleneck in content delivery and sustain huge traffic that is generated by mobile users, specifically for video streaming and content sharing in social networks [2]. Caching popular contents at the edge of the network (at the SBSs) is introduced to reduce the backhaul bottleneck and satisfy user's demand by caching files on-demand upon serving first request for the file. In [2], proactive caching is introduced in small-cell networks by predicting popularity of files based on context-awareness and data mining social network for usage pattern.

In Proactive caching, schemes are developed to predict user's context information and future file usage pattern to cache a content before it is requested [2]. Thus, we can offload the backhaul traffic and download the content at less busy time. It is shown in [2] that proactive caching contents at SBSs significantly reduced the backhaul load and improved user satisfaction.

To estimate the popularity of contents some tools like machine learning or analyzing the infrastructure logs have been studied. Collaborative filtering (CF) methods were applied to predict the popularity. The study showed that exploring user-user relationships through online social networking that strongly affect the network can also assist in storing strategic contents. We discuss their system model below.

The small cell network system presented in [2] includes M SBSs, N UTs (user terminals) and a central scheduler with a limited capacity. The central scheduler is responsible to provide broadband access to SCs over backhaul links. SBSs serve the requests through wireless small cell links or Device-to-Device communications (which will be discussed later), depending on the availability of caches in SBSs or UTs. The goal of this system is to keep the satisfaction ratio above

a threshold by minimizing the usage of the backhaul. It can be achieved by caching the predicted content either at the base-station or at the user terminals.

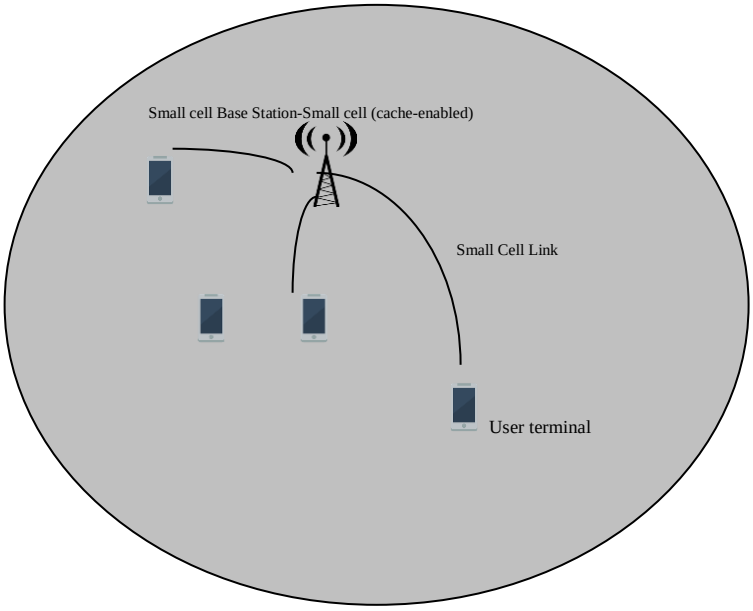

Figure 2. 2 Proactive Caching at Base Stations

Figure 2.2 illustrates the proactive caching at base-stations. The first step is to collect users ranking of the contents even during peak time. Then, by applying CF tools, content popularity are estimated. In the third step, the most popular contents are stored for a given storage size to finally serve the requests from the local cache. The content popularity follows ZipF distribution while arrival times are uniformly sampled in the *T* time duration. Popularity matrix is built for each SBS and a greedy approach is employed to store all popular contents (considering the storage size constrain of SBS). The performance evaluation shows that the proactive caching outperforms reactive caching by improving the satisfaction ratio and reducing the backhaul load.

Content can also be cached at the UTs by exploiting direct communication between devices, known as D2D Communication. To achieve device caching, social ties and physical proximity are two important parameters that play a crucial role in measuring the interactions among users. Influential users (those that have already cached popular contents) offer content delivery through D2D communication. The SBS takes advantage of these users to look for requested content after

it fails to find content in its own cache. If the content is not found in both SBS and device cache of the influential users, then it is finally delivered through expensive backhaul link.

Influential users can be identified by employing centrality metric method in social networks and assessing the quality of their connections. The influential users are the users with higher centrality metric value or with more communication links. Analysis of the content distribution among each social community identifies significant contents of each community that can be stored in the cache of these influential users.

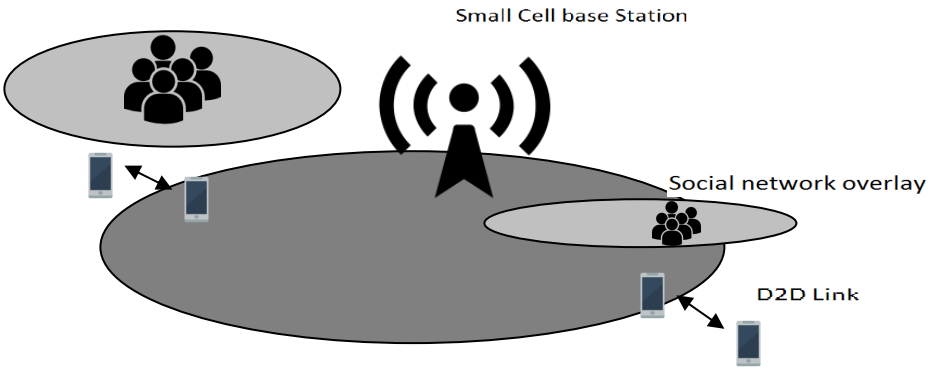

Figure 2. 3 Proactive caching at the user terminal

Figure 2.3 shows the process of proactive caching at user terminals [2]. All users are considered to be in a cluster and this study shows that the proactive approach provides desirable performance in comparison with the on-demand caching.

2.3 Cache Placement for D2D communication in Wireless Networks

One of the significant approach of cache placement in devices was studied in [3]. This approach developed the optimal cache placement scheme in terms of increasing the offloading probability where both the base stations and users have capability of caching. Offloading in their scheme is categorized into three different types: self-offloading, D2D-offloading, and helper-offloading [3]. Figure 2.4 shows the system model of D2D assisted wireless caching system. When a request arrives, each user checks its local cache whether the desired content is stored there to serve

immediately, which is known as *self-offloading*. If the user fails to find the content in its local cache, it launches a search among closest devices. If the content is found in a device cache within a specific radius, that device serves the request and offloading happens through *D2D-offloading*. The helpers- with low rate backhaul link and high storage capacity- if find the content in their caches, transmit the content to the user through *helper-offloading*. Ultimately, if the request couldn't be served through any of the previous offloading stages, the cellular base station is responsible to bring the requested content from the Internet.

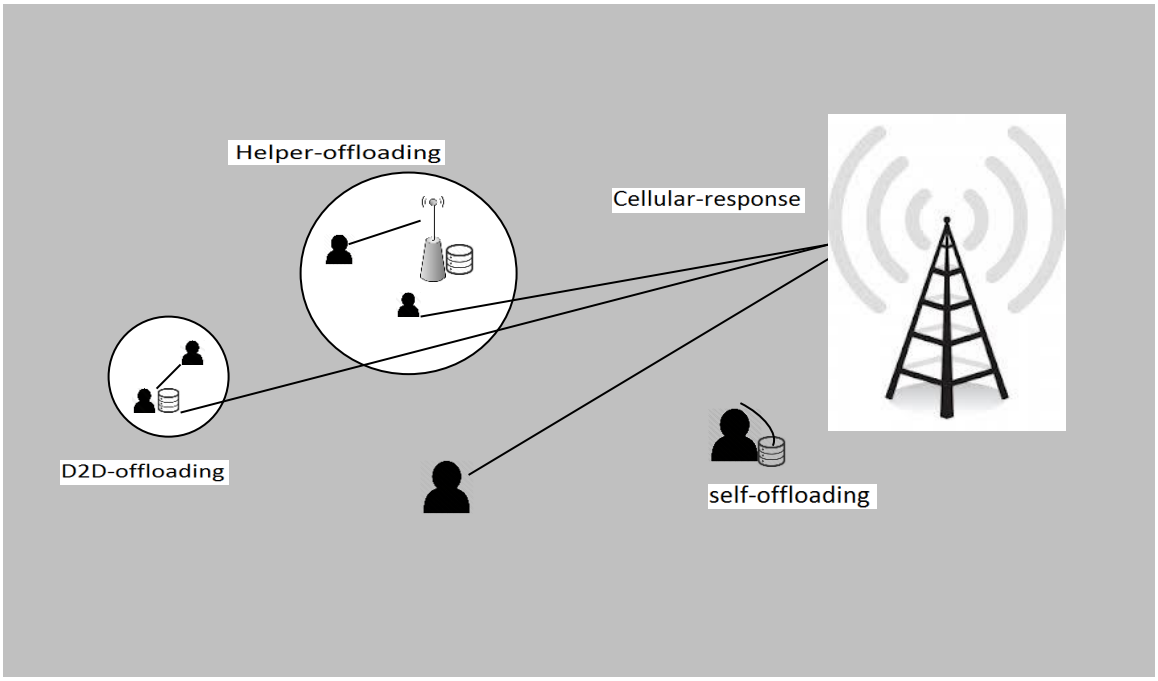

Figure 2. 4 D2D assisted wireless caching system

To reduce the backhaul BS traffic, the scheme in [3] tried to maximize the offloading probability and formulated the optimal cache placement into a difference of convex (DC) problem that can be solved by DC programming. The probability distribution is modeled using PPP distribution with designated density, to find the probability of spreading *n* devices in a definite radius. The paper computes the probability that at least one user caching the *i-th* content, the probability that one helper caching the *i-th* content, the probability that one helper and one user caching *i-th* content, the probability of cached-enabled users, and finally the total offloading probability for D2D assisted wireless caching system. The paper concludes that more data offloading through caching,

lessens the need of transferring the content from the BS. The performance results show that the most popular contents should be cached under low node density but general content can be cached under high node density. Further, the caching scheme in [3] provide a balance among different offloading techniques.

2.4 Collaborative Caching for multi-cell systems

In [4], minimizing the total cost of caching to the content providers through collaborative caching for multi-cell coordinated system are studied. Collaboration among the base stations decreases the operational cost of a network and improves performance. This research considered two types of caching cost: the storage cost and the user attrition (UA) cost. The storage cost is paid by Cellular Network Operator for caching the content. On the other hand, the UA cost should be considered in case of losing users that are not satisfied with the QoS, for example a user experiencing high delay in downloading streaming videos. Therefore, minimizing the cost is tantamount to maximizing users demand satisfaction. This scheme does not consider the popularity of the content instead it employs the concept of competitive ratio to measure the performance of an online algorithm. The algorithm has a potential function for each base station and when a request arrives, the algorithm updates the total UA cost function. It decides to cache the content when the function shows a potential, while caching costs at other base stations exceeded .The problem is formulated as the Integer Linear Program that consists of two cost functions that should be minimized.

The Mobility Management Entity (MME) of a cellular network is responsible for executing the algorithm and the content providers need to pay for running the algorithm at MMEs plus the cost of storing the content at base stations. When the base station receives a request, it immediately serves the request if the content is already cached there. Otherwise, it contacts MME to run the algorithm and decides whether the content should be cached by a new base station or not.

It is shown that the collaborative caching scheme offers greater saving and applying the online algorithms is much more worth than solving non-collaborative optimization problems.

Chapter 3

Design of Deferred Acceptance Placement Algorithm

The goal of this research is to suggest a new caching scheme of dealing with the backhaul problem in wireless access network of femto base stations that have a low-bandwidth backhaul link but high storage capacity. In this network, femto base stations serve as caching helpers that store the most popular files to increase the probability of finding requested files by users within the set of helpers or reducing the downloading time experience by them [1].

These helpers can locally communicate with user terminals (UTs), cache popular files, and serve them in response to user requests. Our proposed scheme differs from previous works in access network caching in two respects: one, we propose an algorithm of optimal placement of files through stable matching theory on bipartite graphs, and two, using a user-to-file association matrix. Knowing a set of files that each user can access, make us rank the files and then place them at the best helper in terms of accessing user and by applying matching game, the most matched files will place in each matched helper within a cluster in small cell network.

In this chapter, we define the problem precisely and outline all possible solutions. Finally, after formulating the problem and outlining its solution space, we present our file-helper placement deferred acceptance algorithm.

3.1 Problem Formulation

In this research, we study a wireless network where the files are requested randomly and they are redundant. They can be requested at a different time based on some popularity distribution. We want to find the best place to cache each file within the helpers in same cluster to increase the probability of finding the requested files among these helpers.

We consider K user terminals (UTs), M helpers and a library of P files connect through a connectivity graph that shows UT k can communicate reliably with helper m. Figure 3.1 shows different components of the network.

When a user sends some requests, one of the local helpers, the one which is directly connect to the user, serves the requests. If the helper couldn't find the requested file in its local cache, neighbor

helper the one which is not directly connect but exists in the same cluster, serve the request otherwise, the macro base station (MBS) handles the request incurring a higher delay.

By predicting the probability of the users' requests and storing them in the helpers' cache, we increase the probability of finding a requested content within the cluster and prevent the backhaul load traffic.

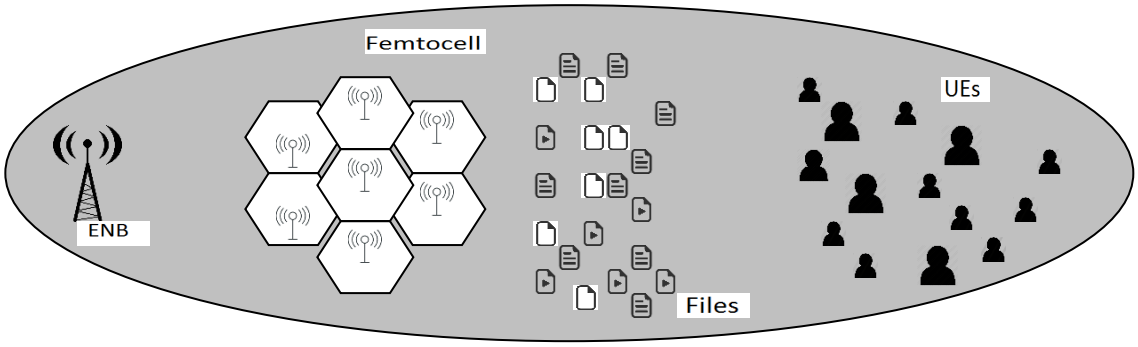

Figure 3. 1 Studied network deployment

We assume that the file download cost per byte from a local helper is C_U, from a helper in the neighbor is C_H, and over the backhaul link is C_B. We further assume following relationship among the three costs:

$$2\ C_H + C_U < C_B \quad (1)$$

It means, if a requested file is found in the local helpers' cache, the download incurs Cu cost, otherwise the file will be downloaded from the local cluster via MBS,and the cost will be $2\ C_H + C_U$. In this case, MBS knows the files are already cached in a helper within the cluster and fetches it to the local helper. All the helpers connected through a single MBS and there is no direct connection between users and MBS. If the file is not with any helper in the cluster, it needs to be downloaded from the network through the backhaul link at the higher $C_B$cost.

3.1.1 Matching Problem

The placement problem is to match the files to the helpers. From the files perspective, it is a one to one matching problem that means each file $p \in P$ should find at most one matching helper to cache it. On the other hand, from helpers perspective, each helper $m \in M$ may find several matching files to cache them as long as the cache capacity q_s allows.

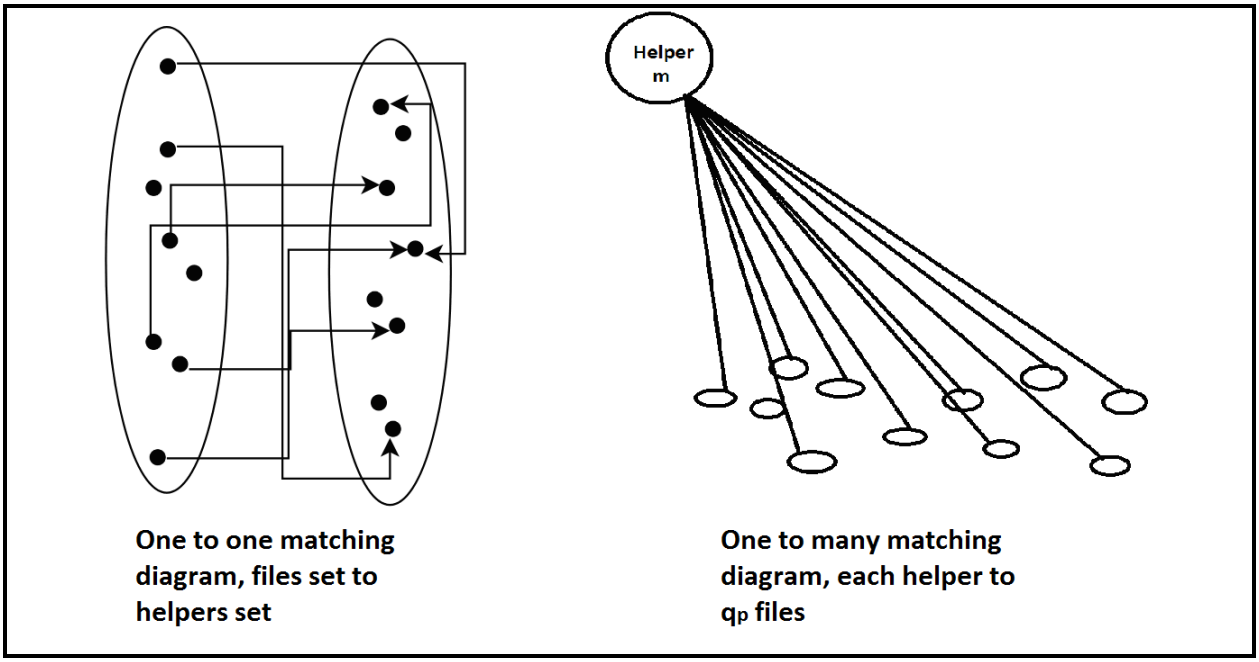

Figure 3. 2 Matching Problem

In particular, our matching problem is to find the most popular files and place them in the most preferred helpers.

It can be shown that there always exists at least one stable matching to:

-Match the files to helpers such that a file is matched to only one helper

-And, a helper matches to several files.

In the following table, we introduce the notations used in our matching game.

Parameter	Description
K= { $k_1 \dots , k_K$}	Set of K users
k_i	Variable indicating i[th] user
M = { $m_1 \dots , m_M$}	Set of M Helpers
m_i	Variable indicating i[th] helper
P = { $p_1 \dots , p_P$}	Set of P Files
p_i	Variable indicating i[th] file
C_U	Download cost from the local helper to the user
C_B	Download cost from the helper in the neighbour
C_H	Download cost from MBS
q_p	Helper *m* can have q_p maximum number of files
q_s	Cache size of each helper
H	Popularity Matrix- Zipf Distribution Modeled
Y	SINR Matrix- user to the helper
Z	File-Helper correlation matrix
A	Connection Matrix-User to Helper
NSize	Normalized file size matrix
StoP	Size of the Files to Popularity Rate matrix
HtoP	Helper to File Association Matrix
B	Matching Matrix

Table 3. 1 List of notations

The popularity of the files in the system is modeled by Zipf distribution popularity where each user ranks each file accordingly. By using Zipf distribution popularity concept we created a user to file correlation matrix to guess the popularity of files ranking by users [7].

$$H_{P,K} = \begin{bmatrix} \alpha_{1,1} & \cdots & \alpha_{1,K} \\ \vdots & \ddots & \vdots \\ \alpha_{P,1} & \cdots & \alpha_{P,K} \end{bmatrix}$$

In $H_{P,K}$ matrix each row shows files from p_1 to p_P while each column represents each user from k_1 to k_K who we have in the system.

Matrix H is a $P \times K$ dimension matrix where each element of the matrix shows the rank of the file corresponding to the row given by the user corresponding to the column. For example, $\alpha_{1,1}$ indicates that file one is ranked by user one with the value α. The highest number of a column shows the highest ranked file for the corresponding user. The average of each row shows the average rank for each file computed over the rank of all users in the system. In [7], the study shows that all these predictions come from the statistical traffic patterns and users' context information (i.e., file popularity distributions, location, velocity and mobility patterns), to allows us to have a

better understanding of users and contents (or files) relation in a network. In our scheme, the matrix H can be derived from any popularity data, such as from recommendation system that often develops user-file association. In this thesis, we assume that the popularity of the files is modeled by a Zipf distribution [9], where the Zipf's skew parameter characterizes the skew in the distribution.

Preferences

In order to define the assignment criteria in matching game aimed to find the most suitable matching of files and helpers [6], we need to find out the preferences of files to helpers as well as helpers to files. Each file can rank one helper in order of its preference based on its local information. In wireless communication, the local information contains channel quality information in terms of SINR value, RSRP even RSRQ or Data Rate [5]. We use SINR value to measure the quality of channel experience by each user connected to each helper.
Matrix $Y_{K\times M}$ gives the SINR value for each helper-user pair, which is computed using the SINR equation [1].

$$Y_{K\times M} = \begin{bmatrix} \theta_{1,1} & \cdots & \theta_{1,M} \\ \vdots & \ddots & \vdots \\ \theta_{K,1} & \cdots & \theta_{K,M} \end{bmatrix}$$

Each row belongs to every user from k_1 to k_K while columns represents helpers from m_1 to m_M. In Y matrix each value $\theta_{i,j}$ represents the value of SINR of *i-th* user to the *j-th* helper. We can drive weighted SINR matrix between files and helpers by multiplying Y with H, the popularity distribution matrix, to get Z = H. Y is a file-helper association matrix. Each $\delta_{i,j}$ is a weighted SINR value of file *i* to helper *j*. In Z, rows show the helpers m_1 to m_M while each column represents every file exists in our system p_1 to p_P.

$$\begin{bmatrix} \delta_{1,1} & \cdots & \delta_{P,M} \\ \vdots & \ddots & \vdots \\ \delta_{P,1} & \cdots & \delta_{P,M} \end{bmatrix} = \begin{bmatrix} \alpha_{1,1} & \cdots & \alpha_{1,K} \\ \vdots & \ddots & \vdots \\ \alpha_{P,1} & \cdots & \alpha_{P,K} \end{bmatrix} \times \begin{bmatrix} \theta_{1,1} & \cdots & \theta_{1,M} \\ \vdots & \ddots & \vdots \\ \theta_{K,1} & \cdots & \theta_{K,M} \end{bmatrix}$$

Denote the channel quality of file $p \varepsilon P$ on helper $m \, \varepsilon \, M$ as $\delta_{P,M} \geq 0$, comes from the matrix Z. Thus, each file has a preference relation $\succ_M$ over the subset of helpers.

From Z, it can be concluded that the high ranked files will likely be cached by the helpers connected to some users through the best quality of channel among all other helpers.

Each helper $m \varepsilon M$ can set its preference for files based on some local information and data coming from file-user association matrix.

We can derive connectivity matrix A from $Y_{K \times M}$, channel quality matrix, by evaluating SINR with the $SINR_{min}$, which is the minimum SINR below that reliable connection cannot be established. The connectivity matrix $A_{K \times M}$ is a Boolean matrix where $a_{ij} = 1$ indicates that connection exists between user i and helper j and $a_{ij} = 0$ indicates otherwise. Each row indicates all connected helpers for the corresponding user and there are K rows for K users. Similarly, each column indicates all the users connected to the corresponding helper and there are M columns for all M helpers.

$$A = \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix}$$

Let us consider $u[m_i]$ be a function that shows the total number of users that are connected to the i^{th} helper m_i, which can be calculated by adding the bits of column m_i:

$$u[m_1] \rightarrow 1+1+0+\ldots+0+1$$
$$u[m_2] \rightarrow 1+1+1+\ldots+0+0$$
$$.$$
$$.$$
$$u[m_M] \rightarrow 1+0+0+\ldots+0+0$$

Where a given helper can be connected to many users and a user to several helpers. Unlike most of the previous works in this area that consider fixed file size with no impact on cache placement decision, we consider file size that varies and becomes a factor in helper's decision of caching a file. To achieve this, we consider matrix $NSize_{P \times P}$, which is a diagonal file size matrix such that

every element $\beta_{i,i}$ along the main diagonal gives the size of the i[th] file. We obtain matrix $StoP_{P\times k}$ by multiplying $NSize_{P\times P}$, and ranking matrix $H_{P\times K}$.

Using $NSize_{P\times P} = \begin{bmatrix} \beta_{1,1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \beta_{P,P} \end{bmatrix}$ and $H_{P,K} = \begin{bmatrix} \alpha_{1,1} & \cdots & \alpha_{1,K} \\ \vdots & \ddots & \vdots \\ \alpha_{P,1} & \cdots & \alpha_{P,K} \end{bmatrix}$

We will have:

$$StoP_{P\times K} = \begin{bmatrix} \omega_{1,1} & \cdots & \omega_{\prime 1,K} \\ \vdots & \ddots & \vdots \\ \omega_{P,1} & \cdots & \omega_{P,K} \end{bmatrix}$$

Where $\omega_{i,j}$ in $StoP_{p\times k}$ matrix, is a weighted file size of file *i* with specific size coupled with its popularity that has been ranked based on Zipf distribution by user *j*. Since all users are not connected with each helper, the popularity used in weighted size should be based on the file popularity assessed by connected users only. Hence, we derive matrix $HtoF_{P\times M}$ where each element $\alpha'_{i,j}$ shows sum of weighted size of i[th] file computed for all users connected to j[th] helper where weighted size is the file size multiplied by user-file association index. Therefore,

$$HtoF_{P\times M} \rightarrow \begin{bmatrix} \alpha'_{1,1} & \cdots & \alpha'_{P,M} \\ \vdots & \ddots & \vdots \\ \alpha'_{P,1} & \cdots & \alpha'_{P,M} \end{bmatrix} = \begin{bmatrix} \omega_{1,1} & \cdots & \omega_{1,k} \\ \vdots & \ddots & \vdots \\ \omega_{P,1} & \cdots & \omega_{P,K} \end{bmatrix} \times \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix}$$

Each row belongs to different helpers m_1 to m_M while each column is for every file in the system p_1 to p_P.

In conclusion, each helper $m \, \varepsilon \, M$ sets its preference for the files in the matrix $HtoF_{P\times M}$ that is a compound value of file size and file popularity evaluated for connected users. Thus, each helper has a preference relation $\succ_P$ over the set of files that creates an ordered list of files $PL_h(m) = \{p_1, p_2 \ldots p_t\}$, which is the files preference list PL_h of *t* most preferred files for helper *m* where

$p_i \succ_P p_{i+1}$ indicates that p_i is more preferred file than p_{i+1} in the list. A helper *m* can select top q_p files from the list of *t* files, which defines the maximum number of files in a helper's cache due to its size constrain. Similarly, each file has a preference relation $\succ_M$ over the set of helpers that creates an ordered list of helpers $PL_f(p) = \{m_1, m_2 \ldots m_s\}$, which is the helpers preference list PL_f that contains *s* most preferred helpers for file *p* where $m_i \succ_M m_{i+1}$ indicates that m_i is more preferred helper than m_{i+1} in the list. Files derive their preference list from matrix Z. In the following we formulate cache placement problem as a graph matching problem.

3.1.2 Matching Sets

A matching η is a function from set $M \cup P$ into the set $M \cup P$. Let us consider $\eta_{(.)}$ is a matching function. If the argument is a file, then $\eta_{(p)}$ gives us a set of matched helpers. If the argument is a helper, then $\eta_{(m)}$ maps to the matched files [5] and we have:

1. $|\eta_{(m)}| = q_p$ for each file $p \varepsilon P$ and if the number of files in $\eta_{(m)} \geq q_p \Rightarrow$ pick high rated file, otherwise check for more user to connect to.
2. $|\eta_{(p)}| = 1$ for every file $p \varepsilon P$, there exist at most one helper $m \, \varepsilon \, M$ that caches the file and $|\eta_{(p)}| = 0$ if $\eta_{(p)} \notin M$.
3. $\eta_{(p)} \in M$ if and only if $p \in \left|\eta_{(m)}\right| \, for \, any \, m \in M$.

For example, matching of files p_1, p_2 and p_3 are $\eta_{(p_1)} = \{m_1 \, m_3\}$, $\eta_{(p_2)} = \{m_2\}$ and $\eta_{(p_3)} = \{\}$ respectively. The above mappings show that file p_1 match with helpers m_1 and m_3 and will be cached in those helpers. Similarly, p_2 finds its matching with m_2 and cached there. In case of file p_3, it is not matched with m_4 due to number of files already cached reaches it limit $q_p = 3$.
The matching matrix $B_{P \times M}$ is as follows:

$$B_{P \times M} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

In the matching matrix, a row corresponds to each file p_1, p_2 and p_3 and a column shows every helper in this matching m_1, m_2, m_3 and m_4. The matching matrix $B_{P\times M}$ is a Boolean matrix where b_{ij} = 1 indicates that matching exists between file p_i and helper m_j and b_{ij} = 0 indicates otherwise. One means specific file and helper are matched while zero otherwise.

3.1.3 Stable Assignment

To show that the matching η is stable according to Gale and Shapley [6] criterion, we review the matching assignment criteria.

A set of *P* files is to be assigned among *M* helpers where the quota of the i^{th} helper in terms of number of files it can cache is q_{pi}.

Each file ranks each helper in order of its preferences, omitting only those helpers it would never accept under any circumstances (admission-college problem [6]). To simplify, we assume there are no ties (indifferent).

Similarly, each helper ranks the files that it wants to cache in order of preferences, by first eliminating those files that cannot be accepted under any circumstances even though the occupied space in the cache is below the quota q_s.

The matching η is blocked by helper *m* and file *p* if helper *m* strictly prefers *p* to some $p' \in \eta_{(m)}$ and p strictly prefers *m* to some $m' \in \eta_{(p)}$

$$\left|\eta_{(p)}\right| \leq q_p \text{ and m is acceptable to } p.$$

Additionally, there always exists one file at least to be matched to one helper based on the stability of *College-Admission* problem [5]:

Among all the helpers in *M* and files in *P*, a stable set of file-helper matching exists in the way that each helper should definitely find its matching file but every helper find top q_p of matched files.

The matching algorithm is modeled as a college admission problem with a straight forward procedure for solving it. In the beginning, all files (students) propose (apply) to their first preferred helper (college), which are the helpers at the top of their helpers preference list.

A helper with a quota of q_p then places on its waiting list the q_p files which ranked highest or all if the capacity is equal or less than q_p and rejects the rest. Rejected files then propose to their

second preferences and again each helper selects the top q_p from all new files and those on its waiting list, will be put on its new waiting list and rejects the rest. The process ends when each file is either on a waiting list or is rejected by every helper to which it is willing and accepted to propose. Now, each helper admits each file on its waiting list and the stable assignment has been achieved.

In [6], the proof is by induction to show that this matching is not even stable but also optimal. It is assumed that to a given point in the procedure, college A hasn't yet decided over it waiting list with quota q_p of better-qualified applicants α_1 to α_i and rejected applicant β . We must show that β is impossible for A while those on the waiting list are possible. We know that every α_i prefer A to all others or it should be rejected by the others, then under our assumption, α_i was impossible for them. Supposed an assignment which β sends to A and everyone else to the colleges they desired to go. In this situation at least one of the α_i should go to a less desirable place than A which show an impossible situation since in this case both A and α_i will upset to the benefit of both. Hence this assignment is unstable and A is impossible for β . Defined procedure shows "Deferred acceptance" procedure which is not only stable but an optimal assignment of applicants.

3.1.4 Proposed Algorithm

The following pseudo-code describes helper deferred acceptance algorithm. In this algorithm files apply for helper based in its preference list. Each helper accepts the application of files applied for the helper in the order of its preference list of files. A helper accepts as many files as the remaining cache size allows.

Algorithm: Find the stable matching for file-helper placement problem (M, K, P, PL, q_p)

Input: M, K, P, PL_h, PL_f, q_p,

Initialization: File=True, Helper=True and $T(p) = P$

While ((File==True)&&(Helper==True))

do

Step 1: Form a list $L = \{(p_1, m_1), (p_2, m_2), \ldots\}$ of (file, helper) pairs formed when each file p_i in $T(p)$ applies for helper m_i such that m_i is the most preferred helper in $PL_f(p_i)$ and $T(p) = \{\}$

Step 2: For every helper m:

1. Form a list $A = \{p_1^m, p_2^m p_3^m \ldots\ldots, p_e^m\}$ that applied for helper m such that $p_i^m \succ_P p_{i+1}^m$ in $PL_h(m)$
2. Form the list by accepting up to $s \leq q_p$ best ranked files: $B(m) = \{p_1^m, p_2^m, p_3^m \ldots\ldots, p_s^m\}$
3. Update both η_f and η_h matching
4. Update the set of rejected files $T(p) = T(p) \cup \{p_{s+1}^m, \ldots\ldots, p_e^m\}$

Step 3: if $(T(p) == \{\})$ then File=False or if no helper is left with less than q_p files in its cache then Helper=False

end

Output: Stable matching η

Table 3. 2 File-Helper proposing Deferred Acceptance Algorithm

Chapter 4

Performance Evaluation

We evaluated the performance of our proposed Helper Deferred Acceptance Algorithm described in chapter 3 using MATLAB. We first discuss the simulation parameters and then analyze the results.

4.1 Experiment Evaluation

We used MATLAB to implement our proposed file-helper deferred acceptance algorithm.

We consider the popularity of content (files) follows Zipf distribution, which is widely used in cache placement studies [1], [9], [18]. In our case, Zipf distribution shows user-file association and provides the ranking of files by a user. We developed the notion of *popularity domain* to reflect different user-ranking of files, which simulates a particular user-file association regime. We used different Zipf parameter (α) to distinguish between popularity domains. We consider two situations: One where all users belong to the same popularity domain following one file ranking simulated by one α-parameter of Zipf distribution. Two, where all users belong to different popularity domain following different α-parameter values of Zipf distribution. We also evaluated how files size impacts placement decisions in terms of the number of files stored in a cache. We consider three file different file sizes; first the situation that size of all files are fixed, second file size has a small variation and finally, file size shows big variation. Thus, we get 6 different combination of these situations, which uses Zipf parameter, $0.4 < skew < 0.9$.

Situation	Description
A1	All users belong to the same popularity domain following one file ranking simulated by one α-parameter of Zipf distribution
A2	All users belong to different popularity domain following different α-parameter values of Zipf distribution
B1	Fixed File Sizes
B2	Small Variation among File Sizes
B3	Big Variation among File Sizes

Table 4. 1Different Situation of effective parameters

Subsequently, we have 6 different scenarios: For $0.4 < \propto < 0.9$;

1. A1 B1
2. A1 B2
3. A1 B3
4. A2 B1
5. A2 B2
6. A2 B3

The capacity of the backhaul links is assumed to be lower than the capacity of the wireless links. The simulation parameters are given the table 4.1.

Parameter	Value	Description
K	*40*	Number of Users
M	*10*	Number of Helpers
P	*150*	Number of Files
SF	*10…100 MB*	Size of the Files
q_s *Cache Size,*	*450 MB*	The Capacity of Each Helper Cache

Table 4. 2 Simulation Parameters

Figure 4.1a, b and c shows the ranking of files following Zipf probability distribution with different skew parameter 0.4, 0.6 and 0.9 in order.

4.1a Skew=0.4 **4.1b Skew=0.6** **4.1c Skew=0.9**

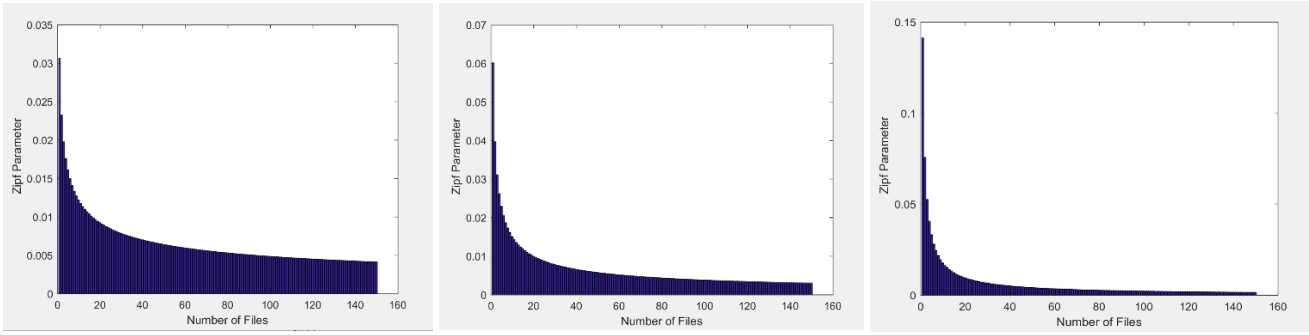

Figure 4. 1a, b and c of Zipf parameter variation in return the number of files

We calculated two metrics to evaluate the performance of our placement algorithm, which is described below:

1. Average Popularity

   We calculate the average popularity of files cached in a helper by taking mean of Zipf probability of all the files cached in that helper. Each file has an associated Zipf probability that is its popularity index. We also calculated Max. and Min. Zipf probability of the cached files.

2. Number of Cached files

   We also computed the number of files cached at each helper that shows the efficiency of cache memory utilization and its sensitivity to the variation in file sizes.

4.2 Results

We discuss results for each of six scenarios below:

Case 1: A1 B1 for skew=0.4

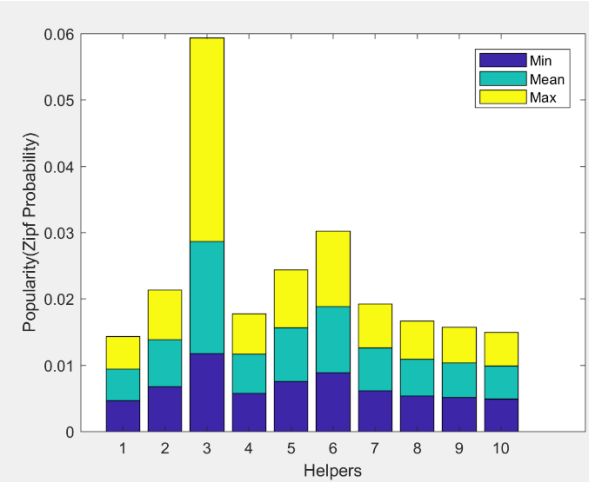

Figure 4. 2a Variation of popularity in return to the helpers

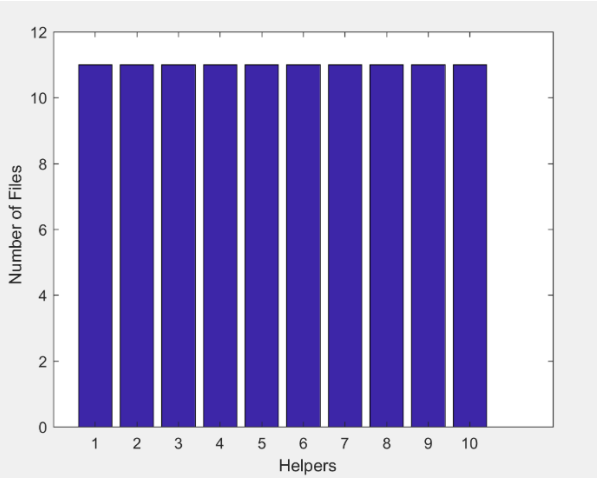

Figure 4.2b Number of files which each helper cached

In the first case, we consider same popularity domain and fixed file sizes. Figure 4.2(a) shows the minimum, mean and maximum popularity of the files cached by each helper. It shows that helper 3 and 6 cached mostly popular files as compared to other helpers.

In our algorithm, since helpers with high preference start caching files, they tend to cache the most popular files. When helpers with low preference get a chance for caching, they tend to get less popular files. As the number of files is more than the capacity of the system, all helpers will cache to their full capacity which is 450, and as the file size is the same, the number of files cached in each helper will be the same. This is shown in Figure 4.2(b). In our case file size equal to 40, so each helper cache 450/40=11 files.

Case 2: A1 B2 for skew=0.4

In this case, we introduce small variation in file size that varies from 30 to 50 with average 40. It causes helpers caching a different number of files.

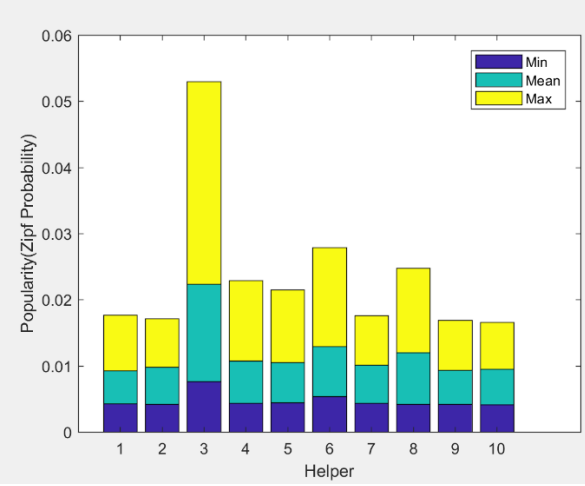

Figure 4. 3a Variation of popularity in return to the helpers

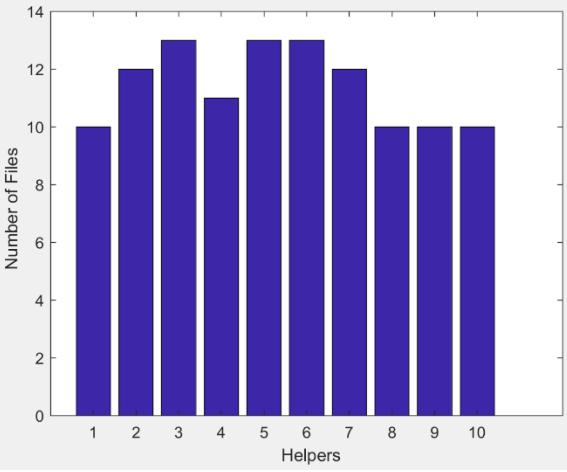

Figure 4.3b Number of files which each helper cached

Case 3: A1 B3 for skew=0.4

In this case, we introduce a large variation in file size, which varies from10 to70 MB. With large variation of file size the number of cached files decreases.

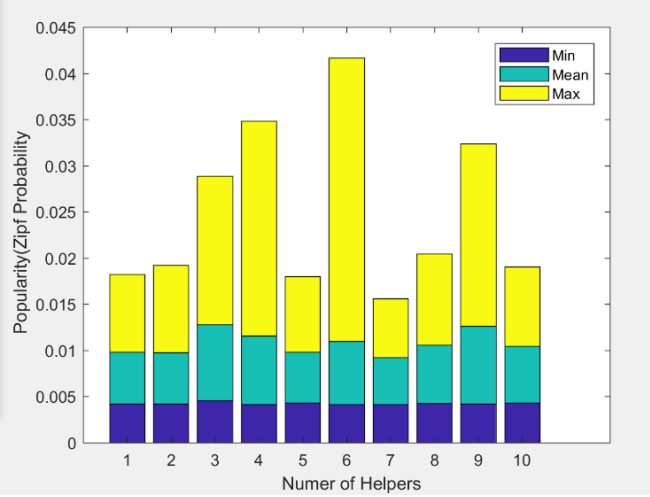

Figure 4. 4a Variation of popularity in return to the helper

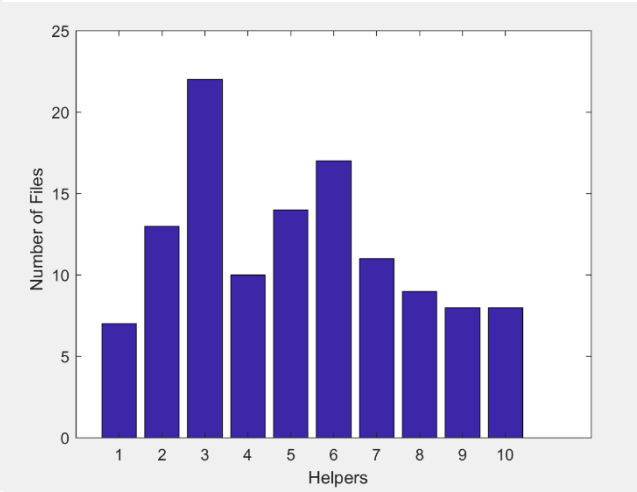

Figure 4.4b Number of files each helper cached

The result shows that the most preferred helpers tend to cache popular files with small size. The preferred helpers are the helpers with the best channel quality (high SINR experienced by users).
In the following cases, we will study how popularity domain influences caching decisions.

Case 4: A2 B1 for skew=0.4

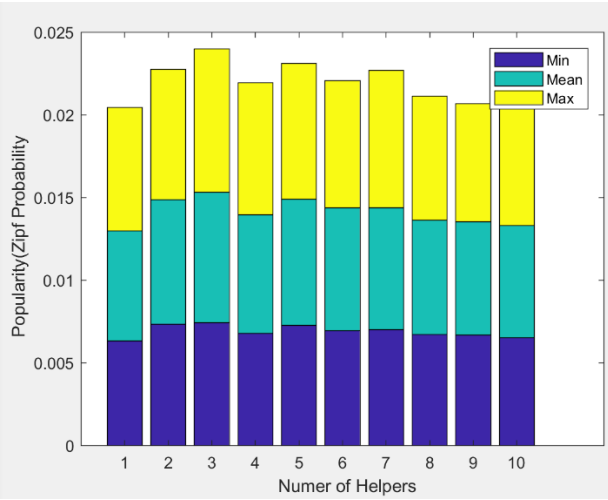

Figure 4. 5a Variation of popularity in return to the helpers

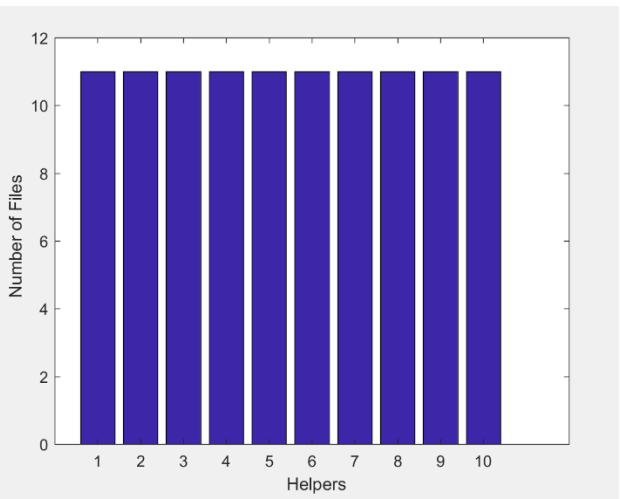

Figure 4.5b Number of cached files for each helper

In the caching decision, a helper with high preference caches files with the highest popularity.

Case 5: A2 B2 for skew=0.4

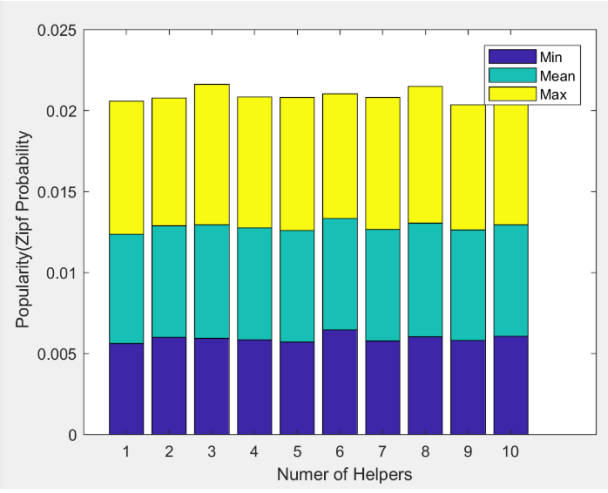

Figure 4. 6a Variation of popularity in return to the helpers

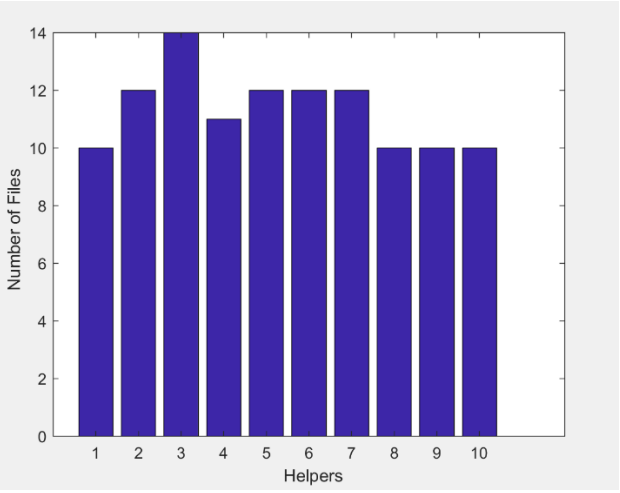

Figure 4.6b Number of files which each helper cached

Case 6: A2 B3 for skew=0.4

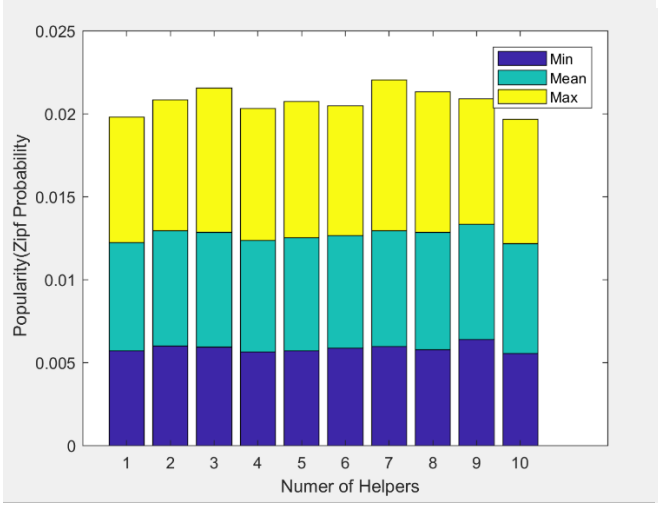

Figure 4. 7a Variation of popularity in return to the helpers

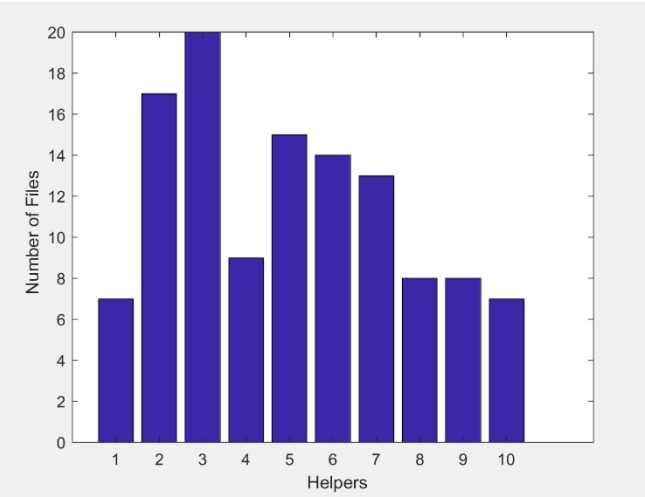

Figure 4.7b Number of files which each helper cached

Condition 4, 5 and 6 prove that when the popularity changes for users, how it moves caching placement and the file sizes, on the other hand, changes both standing to win the matching game pro themselves. In condition 6, each helper cache fewer files in compare when the size is fixed in condition 4.

Case 7: A2 B3 for skew=0.9

In this condition the popularity of users are various and the size of files are between 10 to 70 MB.

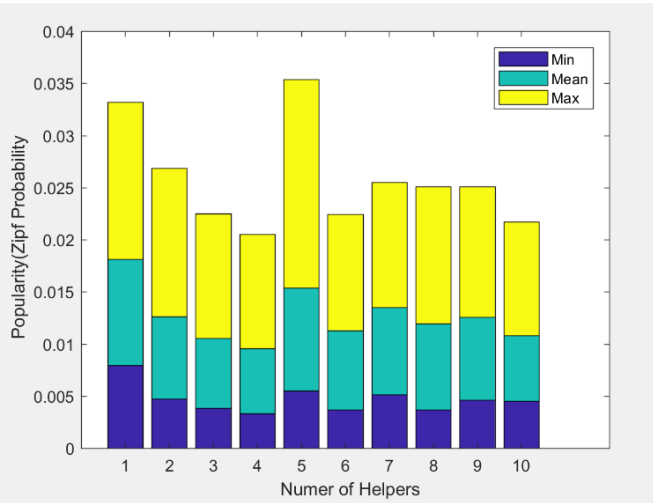

Figure 4. 8a Average popularity in relation between files and helpers

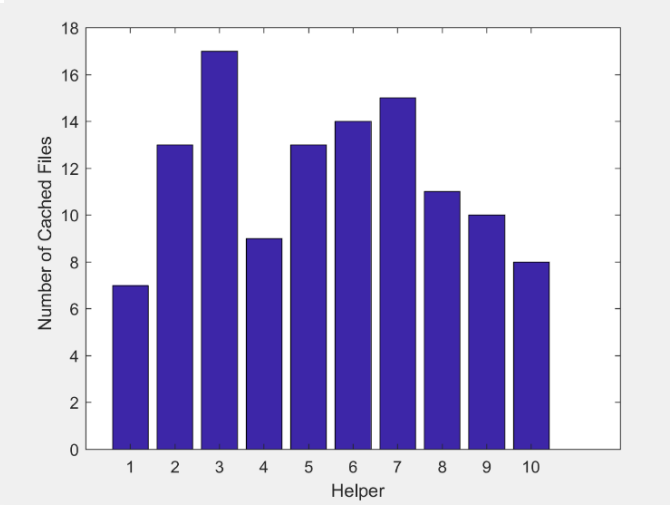

Figure 4.8b Number of files which each helper cached

4.3 Analysis

A1, the popularity distribution of files among users in the same domain, clearly displays that how it makes the popularity rate increases, because the probability of requesting the same file by different user goes up in compare with condition A2 that users are placed in the different domain which they might request totally different files, (A person who likes watching sport game video, might not be interested in fashions matters), but the average of these popularity value seems to be close together.

While the size of the files is fixed (B1) by increasing the skew (from 0.4 to 0.9) the value of Max rises in A1, however, we don't see any significant changes on the value of Max in A2.

When the size of the files change in the small variation (30 to 50M), by increasing the skew value, the Max goes up in A1 but still no big change in A2, therefore, we conclude that in this condition, the popularity distribution changes differently and smallest files with new popularity value, are cached in the most popular helpers.

By growing files' sizes in a big variation, the weight of the size is more than popularity value.(In our matching game the preferences are files with smaller sizes and higher popularity value and helpers with better quality of channel to connect to).

From the whole graph we conclude that: The most popular helpers when the sizes of the files are fixed or have a small variation, cache the most popular file, but when the variation in sizes increases, it affects more in the decision and the most favoured helpers cache files with the smallest sizes rather than the higher value of popularity.

In condition A2, we observed that the value of Min, Mean and Max increases slightly when the skew grows. Popularity doesn't influence a lot.

On the other hand, by having the small variation in sizes for A2, again, most preferred helpers start to cache more files and we can say that the matching game is mostly based on popularity value.

[Big variation as it discussed before, increases the weight of sizes in the decision, by growing the value of skew (skew=0.4 means the number popular files are more than when skew=0.8) even though the Max somewhat goes up but file sizes are dominated].

To have a better result of the way that how placement happens, we required some numerical results as well. By creating a request pattern we were able to observe which files placed on which helper. Then, we could be able to compare our placement algorithm with others.

4.4 File-Helper Deferred Acceptance Placement Algorithm and Random Algorithm

We compared our algorithm with a random placement one to illustrate how much our placement algorithm is optimal. Our users' requests pattern was based on Zipf distribution where skew is equal 0.25, for example, the low value of skew means the number of requests is made by each user tends to be close to each other. In the other hand, the high value of skew means some users will have more chance to generate requests among the others, and each user requests the files based on file popularity distribution.

Rank	File Popularity (skew=0.4)	User1 (file #)	User2 (file #)	User3 (file #)	User4 (file #)	User5 (file #)	User6 (file #)	User7 (file #)	……	User40 (file #)
1	0.0307	13	50	46	60	96	127	42	……..	28
2	0.0232	87	97	142	73	146	45	25	……..	47
3	0.0198	22	4	27	129	26	61	57	……..	122
4	0.0176	60	51	69	26	18	35	101	……..	45
5	0.0161	73	132	95	85	10	119	5	……..	116
6	0.0150	20	78	150	58	78	126	144	……..	48
7	0.0141	6	145	70	17	122	133	89	……..	103
……..	……..	……..	……..	……..	……..	……..	……..	……..	……..	……..
150	0.0041	96	43	23	74	5	26	50	……..	135

Table 4. 3 The files which were requested by each user

Based on this request pattern we defined and calculated 4 different ratios as;

-*LocalHitRatio*, the percentage that local helpers- the helpers which users directly connect to- can serve the requests

-*NeighborHitRatio*, the percentage that those neighbor helpers- the helpers which users do not connect to but they are all connected to one ENB- can serve the request

-*TotalHitRatio*, the percentage that our system can totally serve all coming requests

-*MissRatio*, the number of requests which missed and needed to be served through the backhaul link.

Following table displays the above comparisons;

Parameter	File-Helper Deferred Acceptance Placement Algorithm	Random Algorithm
A1B1-Skew=0.4		
LocalHitRatio	%26	%19
NeighborHitRatio	%69	%32
TotalHitRatio	%95	%51
MissRatio	%5	%49
A1B1-Skew=0.9		
LocalHitRatio	%32	%19
NeighborHitRatio	%44	%32
TotalHitRatio	%76	%51
MissRatio	%24	%49
A1B2-Skew=0.6		
LocalHitRatio	%36	%19
NeighborHitRatio	%46	%32
TotalHitRatio	%82	%51
MissRatio	%18	%49
A1B3-Skew=0.9		
LocalHitRatio	%36	%19
NeighborHitRatio	%58	%32
TotalHitRatio	%94	%51
MissRatio	%6	%49

Table 4. 4 How good it is the File-Helper Algorithm

Our numerical outcome in many different cases with various value for our input parameters prove that the file-helper placement algorithm has a much better result.

Chapter 5

Conclusion

In this study, we introduced a new method for caching placement in small cell networks to increase the probability of finding a requested file by a user in the network. We formulated the cache placement problem as graph matching problem and presented an optimal file-helper matching algorithm based on Gale-Shapley optimal matching criterion [6]. We define stability criterion for the matching and found our matching solution stable in the sense that every helper finds at least one file to cache given no file exceed minimum cache size.

We achieved unique placement of a file within a cluster of helpers to increase the number of files cached within a cluster. We define a cluster as the number of helpers connected through same Node B. Thus, there is no replication of files in a cluster.

Further, our experimental evaluation demonstrates that our algorithm increases local and neighbor hit ratios as compared to a random placement. We also simulated our algorithm for different scenarios by varying file sizes and Zipf popularity parameter. We found that for fixed sized files, helpers tend to cache highly popular files, but for variable sized files, they tend to cache smaller files compromising file popularity ranking. We also found that by applying the file-helper deferred acceptance, we could significantly decrease the traffic that goes over the backhaul bottleneck link.

References

[1] Golrezaei, Negin, Karthikeyan Shanmugam, Alexandros G. Dimakis, Andreas F. Molisch, and Giuseppe Caire. "FemtoCaching: Wireless video content delivery through distributed caching helpers." *2012 Proceedings IEEE INFOCOM*, 2012. doi:10.1109/infcom.2012.6195469.

[2] Baştuğ, Ejder, Mehdi Bennis, and Mérouane Debbah. "Proactive Caching in 5G Small Cell Networks." *Towards 5G*, 2016, 78-98. doi:10.1002/9781118979846.ch6.

[3] Rao, Jun, Hao Feng, Chenchen Yang, Zhiyong Chen, and Bin Xia. "Optimal caching placement for D2D assisted wireless caching networks." *2016 IEEE International Conference on Communications (ICC)*, 2016. doi:10.1109/icc.2016.7511410.

[4] Gharaibeh, Ammar, Abdallah Khreishah, Bo Ji, and Moussa Ayyash. "A Provably Efficient Online Collaborative Caching Algorithm for Multicell-Coordinated Systems." *IEEE Transactions on Mobile Computing* 15, no. 8 (2016), 1863-1876. doi:10.1109/tmc.2015.2474364.

[5] Jorswieck, Eduard A. "Stable matchings for resource allocation in wireless networks." *2011 17th International Conference on Digital Signal Processing (DSP)*, 2011. doi:10.1109/icdsp.2011.6004983.

[6] Gale, D., and L. S. Shapley. "College Admissions and the Stability of Marriage." *The American Mathematical Monthly* 69, no. 1 (1962), 9. doi:10.2307/2312726.

[7] Bastug, Ejder, Mehdi Bennis, and Mérouane Debbah. "Living on the edge: The role of proactive caching in 5G wireless networks." *IEEE Communications Magazine* 52, no. 8 (2014), 82-89. doi:10.1109/mcom.2014.6871674.

[8] Bastug, Ejder, Mehdi Bennis, and Merouane Debbah. "Social and spatial proactive caching for mobile data offloading." *2014 IEEE International Conference on Communications Workshops (ICC)*, 2014. doi:10.1109/iccw.2014.6881261.

[9] Tan, Yuanyuan, Yiling Yuan, Tao Yang, Yuedong Xu, and Bo Hu. "Femtocaching in wireless video networks: Distributed framework based on exact potential game." *2016 IEEE/CIC International Conference on Communications in China (ICCC)*, 2016. doi:10.1109/iccchina.2016.7636817.

[10] Pantisano, Francesco, Mehdi Bennis, Walid Saad, and Merouane Debbah. "Cache-aware user association in backhaul-constrained small cell networks." *2014 12th International*

Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt), 2014. doi:10.1109/wiopt.2014.6850276.

[11] Meng, Yue, Chunxiao Jiang, Lei Xu, Yong Ren, and Zhu Han. "User Association in Heterogeneous Networks: A Social Interaction Approach." *IEEE Transactions on Vehicular Technology* 65, no. 12 (2016), 9982-9993. doi:10.1109/tvt.2016.2525726.

[12] Gu, Yunan, Walid Saad, Mehdi Bennis, Merouane Debbah, and Zhu Han. "Matching theory for future wireless networks: fundamentals and applications." *IEEE Communications Magazine* 53, no. 5 (2015), 52-59. doi:10.1109/mcom.2015.7105641.

[13] Tran, Tuyen X., Parul Pandey, Abolfazl Hajisami, and Dario Pompili. "Collaborative multi-bitrate video caching and processing in Mobile-Edge Computing networks." *2017 13th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, 2017. doi:10.1109/wons.2017.7888772.

[14] Khreishah, Abdallah, Jacob Chakareski, and Ammar Gharaibeh. "Joint Caching, Routing, and Channel Assignment for Collaborative Small-Cell Cellular Networks." *IEEE Journal on Selected Areas in Communications* 34, no. 8 (2016), 2275-2284. doi:10.1109/jsac.2016.2577199.

[15] Hamidouche, Kenza, Walid Saad, and Merouane Debbah. "Many-to-many matching games for proactive social-caching in wireless small cell networks." *2014 12th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, 2014. doi:10.1109/wiopt.2014.6850348.

[16] Ji, Mingyue, Giuseppe Caire, and Andreas F. Molisch. "Wireless Device-to-Device Caching Networks: Basic Principles and System Performance." *IEEE Journal on Selected Areas in Communications* 34, no. 1 (2016), 176-189. doi:10.1109/jsac.2015.2452672.

[17] Pantisano, Francesco, Mehdi Bennis, Walid Saad, and Merouane Debbah. "Match to cache: Joint user association and backhaul allocation in cache-aware small cell networks." *2015 IEEE International Conference on Communications (ICC)*, 2015. doi:10.1109/icc.2015.7248797.

[18] Borst, Sem, Varun Gupta, and Anwar Walid. "Distributed Caching Algorithms for Content Distribution Networks." *2010 Proceedings IEEE INFOCOM*, 2010. doi:10.1109/infcom.2010.5461964.