Technische Universität München
Lehrstuhl für Kommunikationsnetze
Prof. Dr.-Ing. Wolfgang Kellerer

Master‘s Thesis

Game theory based content placement and pricing in connected edge networks

Author:	Banerjee, Anubhab
Address:	Grasmeierstrasse 25
	80805 Munich
	Germany
Matriculation Number:	30681082
Supervisor:	PD Dr.-Ing. habil. Carmen Mas Machuca
Begin:	15. November 2018
End:	14. May 2019

With my signature below, I assert that the work in this thesis has been composed by myself independently and no source materials or aids other than those mentioned in the thesis have been used.

München, 14.05.2019	
Place, Date	Signature

München, 14.05.2019	
Place, Date	Signature

Abstract

Over the last decade, an exponential growth in media traffic calls for designing new network architectures for future content distribution networks. Researchers have focused on mitigating the effect of the media traffic by caching them in in-network nodes closer to the end users. Caching content as supported by Information-centric networking (ICN) is regarded as a very encouraging solution to diminish the backhaul traffic load as well as the latency. Different centrality metrics have been used in the past to model different caching strategies in ICN. In this thesis, we propose a complete operational framework from content provider's perspective which includes placement and pricing of digital media contents. At first we propose a novel caching strategy based on game theoretic centrality metric to store a content close (here 'close' means that minimum number of hops are required by the user(s) to locate and get the content in a sparsely connected, multi-hop network) to the potentially interested user(s). Based on the content access history of BBC iPlayer in UK, we evaluate our proposed content placement solution in two different real life scenarios. The evaluation is performed to measure different metrics like traffic savings, energy savings, etc. It is also shown that the performance our proposed solution outperforms the state-of-the-art centrality measures. Along with the content placement, we provide a content pricing policy also from the content provider's perspective. Using Nash Bargaining Solution (NBS), we propose a pricing policy so that the both users and platform provider maximize benefits, while the load of the server is minimized.

Contents

Chapter 1

Introduction

1.1 Peer-to-peer content sharing architecture

Throughout the decades media industry is shifting from delivering content via physical medium to delivering content via digital medium. Primarily, there are two types of digital content distribution models: centralized and decentralized:

- *Centralized model* : This model is implemented as a client-server system where content placement, content pricing and customer relationships are centrally managed by the content provider. In this model, a user can get his/her desired content from the server of the content provider only.
- *Decentralized model* : A decentralized model can be implemented as a peer-to-peer (P2P) networks (like *Wi-Stitch*) where the users participate into the process of content distribution by trading content [1]. The user can store the content in his cache-enabled home devices (like, Apple TV) after downloading it from the server of the content provider and share it with other users who are interested in the same content.

After the commercialization of internet in 1994, there has been an exponential growth in digital media content delivery services [2]. According to a CISCO report [3], almost 84% of the internet traffic will be video content by the end of 2019. Over recent years researchers have focused on mitigating the effect of the media traffic by caching them in in-network nodes [4]. While dealing with the same problem, we seek a different, more sustainable approach that makes use of *existing* resources more effectively. In particular, we note that current versions of Wi-Fi such as 802.11ac and 802.11n have speeds reaching over 1.5 Gbps within close proximity. These speeds far exceed the commonly available broadband connection speeds in developed countries today [1] suggesting that it is possible to efficiently

[1]For example, in the UK, the average home broadband speed in 2017 was 46.2 Mbps https://www.ofcom.org.uk/research-and-data/telecoms-research/broadband-research/home-broadband-performance-2017

distribute content by *sharing content at the edge.*

A content can be cached at various levels of the intermediate nodes between the server and the requester of the content. The advantages of keeping the content closer to the server are:

- more number of user requests can be satisfied with the same content,
- number of duplicate contents in the network decreases, and
- less amount of storage space is required to maintain the network operational.

On the other hand, keeping the content closer to the users helps in reducing

- backhaul traffic load, and
- latency [5].

Based on the two ideas mentioned above, Raman *et al.* [6] recently introduced the *Wi-Stitch* architecture, showing the feasibility of sharing content at the edge of the network. Wi-Stitch proposes to stitch together a Content Delivery Network (CDN) by linking together users' homes using Wi-Fi (or similar technology), creating a local cell of users who can share content among each other. This sharing is enabled by provisioning storage for the shared content at users' homes. We envision that the cache would be attached to, e.g., the ISP's modem in the user's home, or a media streaming device operated by the content provider (some, such as versions of Google Chromecast and Apple TV, already come with attached storage, but need to be interconnected with neighbours).

The possible savings in the Wi-Stitch architecture depend on two factors: *(i)* the number of users who are within Wi-Fi range of each other, and *(ii)* the amount of similarity, or shared interests, in their content access patterns. Using a dataset consisting of accesses to a national TV streaming service in the UK, Raman *et al.* [6] showed that significant savings can be achieved, even if users simply share content that they have previously accessed with everyone else within their cell.

1.2 Challenges of this architecture

In Information Centric Networking (ICN) content can be cached at any level between the server and the user [4]. That is why in ICN several caching policies [7, 8] have been proposed depending on fulfilling different requirements. The main difference between *Wi-Stitch* architecture and ICN is that in *Wi-Stitch*, contents are cached only at the very edge, i.e., only at the users' homes. In this type of connected edge architecture for content sharing, two foremost concerns are *what* and *where* to cache the contents. These problems are described below in detail:

Parameter	Value
Number of users	21 millions
Number of IPs	16 millions
Number of sessions	231 millions

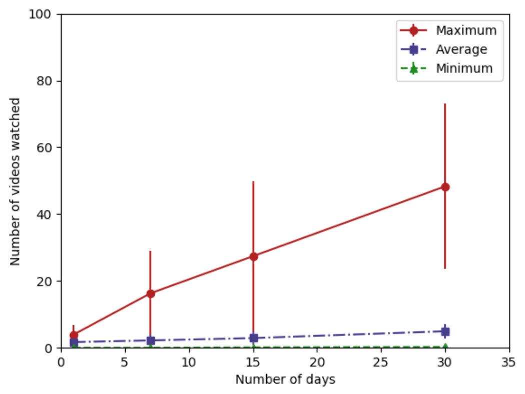
Table 1.1: BBC iPlayer dataset values

- The size of the content universe gets larger day by day, but a user has only a certain, finite and limited storage capacity at home. Naturally a user cannot cache all the content and can only store a selected number of contents at home devices. Thus, whenever a new content arrives, the user should be able to make a decision whether to cache it or not.
- Similarly, for a very popular content, it is difficult to put it closer to the interested user(s) and find one particular device where the content can be cached so that it can be shared by maximum number of interested users with minimum number of hops, thus reducing the latency and resource utilization.

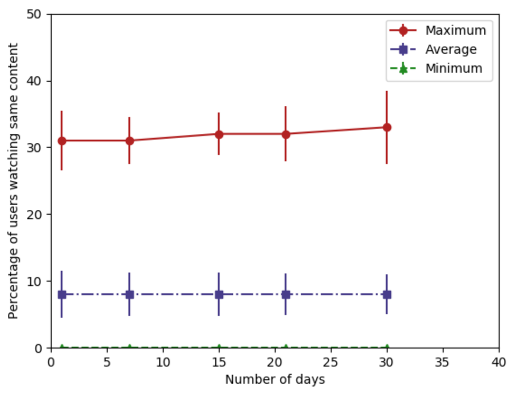
1.3 Motivation and thesis objective

Let us start with the motivation for the addressed problem by using a real life dataset of the BBC iPlayer, which is one of the most popular catch-up TV services in the UK, used by over 60% of the total population. We use the BBC iPlayer trace file of July 2014 which consists of around 231 million sessions from 21 million distinct users and 16 million different IP addresses.

Fig. 1.1a depicts the cumulative number of watched videos over time (so-called watching rate) in 100 sets of 1000 users chosen randomly from the dataset without replacement. It can be observed that in a month, a person watches up to 50 videos and a mean of 5 videos on average. Furthermore, it can be observed the percentage of the users watching the same video is quite high and stable over time as depicted in Fig. 1.1b. It can be observed that during one month, same videos have been watched by 8 different users on an average, and up to 32 different users at maximum. Some of these users, if not all, happen to live in the same locality and could share content among them using their Wi-Fi. In this way, instead of getting the same content from the server repeatedly, the already existing Wi-Fi network can be used; and consequently, the latency and network traffic will significantly be reduced. In this case, the problem we want to address is *given a distribution of users, their connectivity and their expected interest to different contents (based on their history), find the best location of each content so that the number of hops from any user to its content is minimized and the sharing factor of the same content is maximized.* The solution will then minimize the network traffic as well as the latency.



(a) BBC iPlayer watching rate in UK



(b) Similarity of watching

Figure 1.1: iPlayer watching UK

1.4 Our contributions

In this thesis, we provide a complete operational framework from a content provider's perspective. Our main contributions are two fold:

- **Content placement** : We select the most suitable user(s) where the content provider should place a content such that traffic savings are maximizied and latency is minimized. In Section 1.4.1 we have discussed our contributions in this scope in more detail.
- **Content pricing** : For this kind of peer-to-peer content sharing architecture, we also provide a pricing policy on the contents from the content provider's perspective. We have described in more detail our contributions in this scope in Section 1.4.2 .

A user can select either of these two options (centralized and decentralized) to get a content. For the motivation purpose of the content placement problem, we assume that all the users are participating into content sharing. In the content pricing problem, we discuss all possible circumstances and how a content should be priced under different circumstances.

1.4.1 Content placement

In the scope of the content placement problem, we can summarize the main contributions of this thesis as -

- We utilize *game theoretic centrality* to place contents close to the interested user(s) in *Wi-Stitch* kind of architecture. To the best of our knowledge, we are the first ones to use game theoretic centrality in the study of a content placement problem.

- We create a graph where the nodes are users' homes and an edge between any two nodes signifies that they can be connected with each other via Wi-Fi and are capable of sharing content. We find the most influential node in the graph where a content can be cached so that it can be shared by maximum number of nodes with minimum number of hops. We find this most influential user in the network using different centrality measures (degree centrality, betweenness centrality, eigenvector centrality, closeness centrality, game theoretic centrality) and give a comparison of their performance for different cases. Based on these results we find that choosing the most influential node using game theoretic centrality is the best option for content placement and sharing problem. This work has been accepted in the conference ICTON 2019 [9].
- We discuss the scalability limitations of calculating game theoretic centrality for large number of users and propose non-optimal alternative methods of computation to reduce the computation time. We also compare the advantages and disadvantages of these methods.
- To evaluate our proposed strategy in a more realistic scenario, we consider a single street with square shaped buildings on both sides of it laying on both sides of the street. We consider two different kinds of scenarios -

  1. *singlehouse model* : This model considers each building as a single house and there is a router in each house.
  2. *apartment model* : Here in each building there are several floors and in each floor there is an apartment. There is a router in each apartment.

  In both models, the routers act as nodes in the graph. After a very detailed pathloss model analysis [10, 11], if the link capacity between considered routers is above a certain threshold value, we create an edge between them (Step "Graph generation" in Section 3.2). After the edge network is created, we place the content in the designated nodes using our proposed approach (Step "Content placement" in Section 3.2) and measure all the necessary parameters (Step "Evaluation" in Section 3.2). We randomly select n number of different IP addresses from the BBC iPlayer dataset and assign each of them randomly to one of the routers. We study the amount of traffic savings, energy savings, etc. we are getting under different circumstances and also measure how closely our strategy has been able to bring content to the interested user(s).
- When the number of nodes are relatively high in the network (i.e.,in a dense urban scenario), a single copy of content may not be enough if multiple users want it and they are unable to share the content among themselves. Hence, the same content has to be placed multiple times. We propose four methods for multiple content placement and compare their performances to show which placement method is suitable for which kind of scenario. Finally their performances under different scenarios are evaluated.

1.4.2 Content pricing

In the scope of the content pricing, we can summarize our contributions as -

- Depending on the number of users who are participating into content sharing, we discuss three possible cases and using Nash Bargaining Solution (NBS) [12], we calculate the optimal price in each of these cases.
- We also consider another factor from content provider's perspective, which is the reduction of load on the server. Taking this factor and generated revenue both into account, we calculate an equilibrium price of the content which is not disturbed by joining or removal of any number of users.

To the best of our knowledge, we are the first ones to introduce an NBS based pricing policy for such kind of hybrid digital media delivery system. This work has been submitted to CTTE-FITCE 2019 [13].

1.5 Thesis organization

The rest of the thesis is organized as follows. **Chapter 2** presents a brief background on the existing caching mechanisms and peer-to-peer content sharing architectures along with discussion on how state-of-the-art centrality measures had been used in the past to deal with the content placement problem. Along with that, we also introduce the game theoretic centrality and the efficient ways to calculate the centrality values for larger networks. In **Chapter 3** we formulate the content placement problem, define the payoff function that suits our need and provide a complete content placement framework. As a part of the framework, we also discuss on user clustering and different graph partitioning algorithms. In **Chapter 4** we evaluate the performance of our proposed strategy in two different real life scenarios. In **Chapter 5** we propose a content pricing policy from the content provider's perspective in this kind of content sharing architecture with concluding the thesis at **Section 6**.

Chapter 2

Background and introduction to game theoretic centrality

2.1 Existing content placement strategies

Throughout the years researchers have focused on how to mitigate the effect of the media traffic which had an exponential growth over the last decade by caching them in in-network nodes [4, 14]. They tried to use the advantages of content caching in wireless networks to reduce traffic load and showed that reasonable offloading gains can be obtained. In [15] the authors provided an overview of throughput and outage scaling of wireless networks with caching. They also compared the device to device communication approach with other approaches (like conventional unicasting, harmonic broadcasting) based on combinatorial cache design. In [16] advantages of caching popular videos in the storage of small-cell base stations were discussed. The researchers tried to cache videos inside small cells to make downloading videos easy for the users which in turn reduced traffic load and latency. In [4] the authors presented presented a comprehensive survey of state-of-the-art caching techniques in Information Centric Networking (ICN), new features of ICN caching and some optimization mechanisms like cache dimensioning, cache sharing and so on. Authors of [17] designed a caching strategy in socially-aware D2D communication by jointly considering social ties, common interests, and the D2D transmission scheme. These caching policies are designed when the content can be cached at any node between the server and the content requester (user). In this thesis we present a content placement policy where the content is cached at users' homes only.

In a network, caching can be done in two ways - proactively or reactively. In ICN caching is done reactively - whenever a user requests for a content for the first time, the user gets it directly from content custodian and caches it so that other users in the network can use the same content until the content gets kicked out following the underlying content replacement algorithm (e.g. LRU). But reactive caching has certain limitations. In [18] the authors discussed the limitations of reactive caching in networks. That is why in this thesis we consider a proactive caching approach.

In [18] the authors proposed a novel proactive caching paradigm that deal with dynamic aspects of context-awareness and social networks. They also studied two particular cases where proactive caching plays an important role and showed that a high amount of backhaul savings and user satisfaction ratio can be obtained by taking their approach. Researchers in [19] proposed a proactive caching architecture for 5G wireless networks by processing a large amount of data on a big data platform and determine the potential gains of these techniques for cache-enabled wireless networks. One of the most important parameters in proactive caching is user preference. A group of researchers proposed a supervised machine learning model that learns from user preference [20]. They tried to preload the mobile device with the content generated by this learning algorithm and showed that it can save a huge amount of mobile data. In [21] it was claimed that content popularity and user preferences are two key features of content delivery traffic also. The work in [22] studied the impact of proactive resource allocation exploiting the predictability of user behavior for load balancing.

The works we have discussed so far, proposed several proactive caching strategies based on user preferences aiming to reduce traffic consumption. In this paper, we propose a caching strategy that reduces traffic consumption as well as minimizes latency by reducing average number of hops. Our proposed strategy based on game theoretic centrality minimizes latency by putting the content close to the interested users(s) so that a content is available to a user within a few hops.

2.2 Centrality measures in content placement

Some of the existing solution in the state-of-the-art address the content placement problem based on the network topology characteristics [7, 23]. In particular, they are based on centrality measures, which help to identify the most "important" node in a network. This section gives an overview of the most relevant centrality measures considered for content placement problem.

Let us represent a network as a graph $G = (N, E)$ where E and N denote the edges and nodes of the network, respectively. An edge between two nodes n_i and n_j is represented by $e_{ij} \in E$.

Node Degree Centrality The concept of node degree centrality was first introduced in [24]. The basic working principle behind this centrality is - the more connected a node is, the more important it is in the network. The *normalized degree centrality* of a node n_i

can be expressed as -

$$\mathbf{C}_{degree}(i) = \frac{x(i)}{N-1} \tag{2.1}$$

where $x(i)$ denotes the node degree of n_i, which is the number of nodes in the network with which node n_i is directly connected.

Betweenness Centrality The idea of betweenness centrality was first discussed in [25]. The working principle behind this centrality is - the more information flows through a node, the more importance the node has in the network. The information flow can be measured by counting the number of shortest path between any two nodes in the network that passes through the node under consideration. Let sp_{ij} denote the number of shortest paths between nodes n_i and n_j in the network. Let us also assume that out of these paths, sp_{ikj} number of shortest paths passes through the node n_k. The *normalized betweenness centrality* of node n_k can be given by

$$\mathbf{C}_{betweenness}(i) = \frac{2}{(N-1)(N-2)} \cdot \sum_{i,j \in N \setminus k} \frac{sp_{ikj}}{sp_{ij}} \tag{2.2}$$

Eigenvector Centrality The eigenvector centrality [26] measures the centrality of a node based on the centrality of its neighbors. If $A = [a_{v,t}]$ denotes the adjacency matrix of G (i.e. if nodes n_i and n_j are connected, $a_{ij} = 1$, otherwise $a_{ij} = 0$), then the *relative centrality score* of node n_i is

$$\mathbf{C}_{eigenvector}(n_i) = \frac{1}{\lambda} \cdot \sum_{j \in G} a_{ij} \mathbf{C}_{eigenvector}(n_j) \tag{2.3}$$

where λ is a constant. This can be rewritten as

$$A \cdot \mathbf{C} = \lambda \cdot \mathbf{C} \tag{2.4}$$

where λ is the eigenvalue.

Closeness Centrality The closeness centrality [27] is the reciprocal of the sum of the shortest path distances from a node to all other nodes. In other words, it is a quantification of the distance of a node to all other nodes. The *normalized closeness centrality* of node n_i can be given by

$$\mathbf{C}_{closeness}(i) = \frac{N-1}{\sum_{j \in N} l_{ij}} \tag{2.5}$$

where l_{ij} represents the shortest distance between nodes n_i and n_j.

Let us consider the example graph depicted in Fig. 2.1. For each node, these four types of centrality values have been calculated and the results are shown in Table 2.1. All results

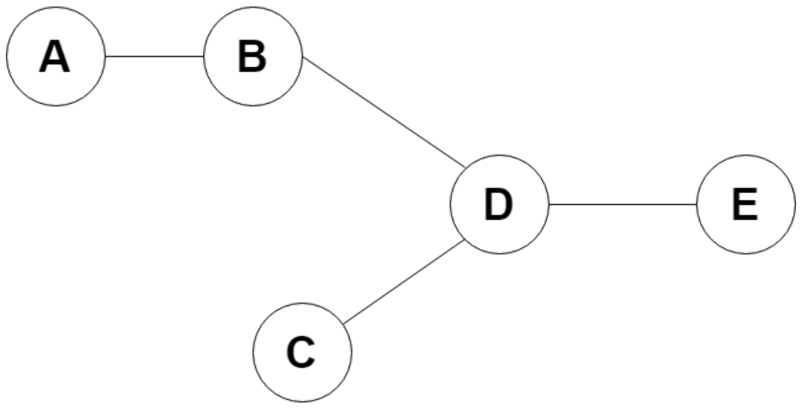


Figure 2.1: Example network

Node	Node Degree	Betweenness	Eigenvector	Closeness
A	0.25	0.0	0.271	0.444
B	0.5	0.5	0.5	0.667
C	0.25	0.0	0.354	0.5
D	**0.75**	**0.833**	**0.653**	**0.8**
E	0.25	0.0	0.354	0.5

Table 2.1: Centrality values for all nodes of example network in Fig. 2.1

have been approximated to 3 decimal places only. In each category, D is the node with the highest centrality value.

In [23] the authors explored the advantages of centrality-based caching in SBSs (Small Base Station) scenario and compared performance results among four centrality measures - degree, closeness, betweenness and eigenvector. In [7] it was demonstrated that selective instead of ubiquitous caching can yield higher gain. The authors of this paper also proposed a caching scheme based on betweenness centrality and showed by simulation results that their solution can achieve better gain both in synthetic and real topologies. In [28] the authors proposed a new community detection inspired proactive caching scheme for device-to-device (D2D) enabled networks and used Eigenvector Centrality measure to locate the influential users in the community structure. But in none of the works user preference was taken into account. In this paper we use game theoretic centrality [29] to find the best user in a network for content placement. To the best of our knowledge, we are the first ones to design a caching strategy based on game theoretic centrality that takes user preferences into account.

2.3 Introduction to game theoretic centrality

Let us revisit some basic definitions in game theory before we introduce our proposed game theoretic centrality. A *normal form game* (for a more detailed understanding, please refer to [30]) consists of three components -

- A set of players $N = \{1, 2, ..., n\}$.
- A set of pure strategies $\{S_1, S_2, ..., S_n\}$.
- A set of payoff functions $\{pf_1, pf_2, ..., pf_n\}$ each assigning a payoff value to each combination of chosen strategies.

In a cooperative game, the players try to come to an agreement, and they also have a choice to bargain with each other, so that they can gain maximum benefit. This benefit is higher than what they could have obtained by playing the game without cooperation.

In a cooperative game, any number of players can form a coalition S. A payoff function pf can be defined to assign an award value $pf(S)$ to each coalition. The importance of each node in the coalition is measured by calculating the Shapley Value [31]. For each player, this value is calculated in each possible coalition the sum of all these values tells us how influential the player is. For any player i, $i \in \{1, 2, ..., n\}$, the Shapley value can be calculated as

$$\zeta_i(pf) = \sum_{S \subseteq N, i \notin S} \frac{|S|!\,(|N|-1-|S|)!}{|N|!} \cdot [pf(S \cup \{i\}) - pf(S)] \tag{2.6}$$

The game theoretic centrality is then defined as

$$\mathbf{C}_{game-theoretic}(i) = \zeta_i(pf) \tag{2.7}$$

Let us apply this metric to the network of Fig. 2.1. Let the nodes $N = \{A, B, C, D, E\}$ denote users that watch videos from the same streaming service (e.g., Netflix, BBC iPlayer). Let us assume that they are in close proximity of one another, so that they are able to have Wi-Fi connectivity as shown in the figure with edges $E = \{e_{AB}, e_{BD}, e_{CD}, e_{DE}\}$. A user n_i can only share content via Wi-Fi with another user n_j if there exists an edge e_{ij} between them. In real life, the possibility of existence of an edge depends on the distance between the users as well as the available bandwidth. All the nodes can have the strategy of forming a coalition S by cooperating with other nodes, which results in a subgraph formed by the nodes retaining the same properties from the original graph. A coalition can be formed by any number of nodes in the graph. In our example, the possible coalitions are $\{\{\emptyset\}$,$\{A\}$, $\{B\}$, $\{C\}$, $\{D\}$, $\{E\}$, $\{A, B\}$, $\{A, C\}$, $\{A, D\}$, $\{A, E\}$, $\{B, C\}$, $\{B, D\}$, ..., $N\}$. The payoff function pf that assigns an award value $pf(S)$ to each coalition possible in N is given by the game policy (e.g., giving award for sharing its content and/or for using its own storage capacity for other content). The payoff for an absence of coalition $pf(\emptyset)$ is 0. Based on this payoff function, the Shapley value of each player is then calculated using Eq. 2.6. The node with the highest Shapley value will be considered as the most influential node in the network.

2.4 Efficient computation of Shapley value

The biggest disadvantage of Shapley value is its high computation complexity. When the value of $|N|$ is quite high (≥ 50), it takes a huge amount of time for the computation of Shapley value. To circumvent this problem, over the years various approaches have been proposed for efficient computation of Shapley value. Some of them are enlisted below -

- 1. Multi-Linear Extension Method (MLE) [32]
- 2. Modified MLE Method [33]
- 3. Random permutation Method [34]
- 4. **Monte carlo Simulation method** [35]
- 5. **Fatima's Linear Approximation method** [36]

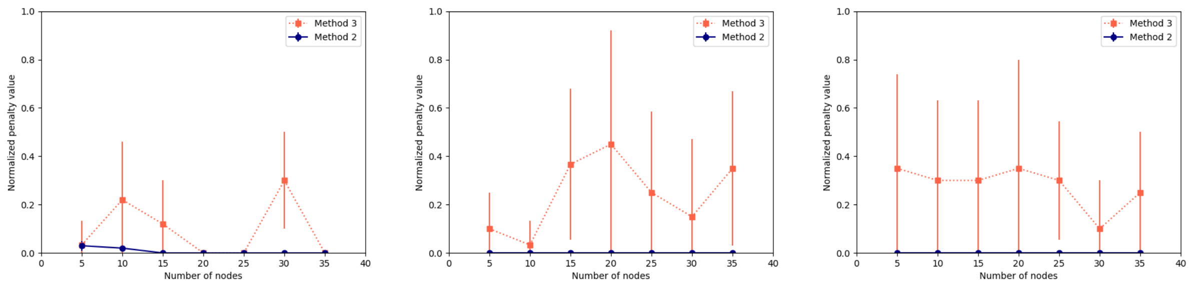
It is to be noted that not all these methods can be applied to calculate Shapley value in any kind of scenario. Out of the five methods mentioned above, only [35] and [36] can be applied to calculate Shapley value for nodes in a graph. In this section we compare the performance between these two methods and select the one most suitable for us.

We generate graphs according to Erdos-Renyi model [37] for various N (the number of nodes in the graph) and p' (probability that an edge exists between any two nodes in the graph) values. For lower values of p', the graph is sparsely connected and for higher values of p', the graph is densely connected. We assign a unit weight to every edge and randomly assign weights between 0 and 1 to each of the nodes in the graph. After that calculate game theoretic central node using the original method described in Section 2.3 and the methods we want to compare between ([35] and [36]). For easier understanding, we denote the method described in Section 2.3 as Method 1, the method described in [35] as Method 2 and the method described in [36] as Method 3. [1] The simulations are run on a 8-core Intel Xeon CPU with processor speed 2.8GHz and 24 GB RAM and during our simulations, we vary N between 5 - 100 and p' between 0.2 - 0.8.

We measure the following two parameters -

- **Penalty Values** : Using Method 1 we always get the correct game theoretic central node. While using Method 2 or Method 3, if the calculated game theoretic central node is same as the one we get from Method 1, then the solution is correct and no penalty is added. Otherwise, a penalty value equal to the number of hops between

[1]The python code to calculate Shapley values using these three methods is available here: http://tiny.cc/1qi75y.



(a) p'=0.2 (b) p'=0.5 (c) p'=0.8

Figure 2.2: Penalty values of Method 2 and Method 3 w.r.t. Method 1

the node calculated by Method 1 and the node calculated by Method 2 and 3 is added. The penalty value has been normalized with respect to the diameter of the graph.

- **Execution Time** : In each run, we measure the total time consumed by each of these methods to calculate the Shapley values and to find the game-theoretic central node.

For each particular (N, p') value, we generate a graph and we run the simulations 20 times, each time assigning new weights to the nodes randomly and measure the parameters mentioned above.

Penalty values From Fig. 2.2 we see that in Method 2 penalty value is almost always 0 except for a particular case $(N = 5, p' = 0.2)$. For Method 3, the penalty value is non zero most of the time and this value is higher when the value of p' gets higher.

When $N \geq 35$, the time and space complexity of Method 1 becomes huge and it becomes impossible to calculate game theoretic central node. Based on Fig. 2.2 we can assume that the penalty value using Method 2 will always be 0 for any (N, p') combinations when $(N \geq 10)$. That is why for higher values of N, we measure the penalty values of Method 3 w.r.t. Method 2 instead of w.r.t. Method 1 as depicted in Fig. 2.2.

The result is shown in Fig. 2.3. We see that after $N = 50$, for all p' values the penalty values tend towards to a constant value equals to 0.5.

Execution time Time consumed by Method 2 and Method 3 under different circumstances is given in Table 2.2. We can see that Method 3 is much much faster than Method 2. For example, when the number of nodes is quite high (e.g. $N = 80$), calculating game theoretic centrality node takes 30-40 minutes for Method 2, while for Method 3 it only takes 5-10 seconds.

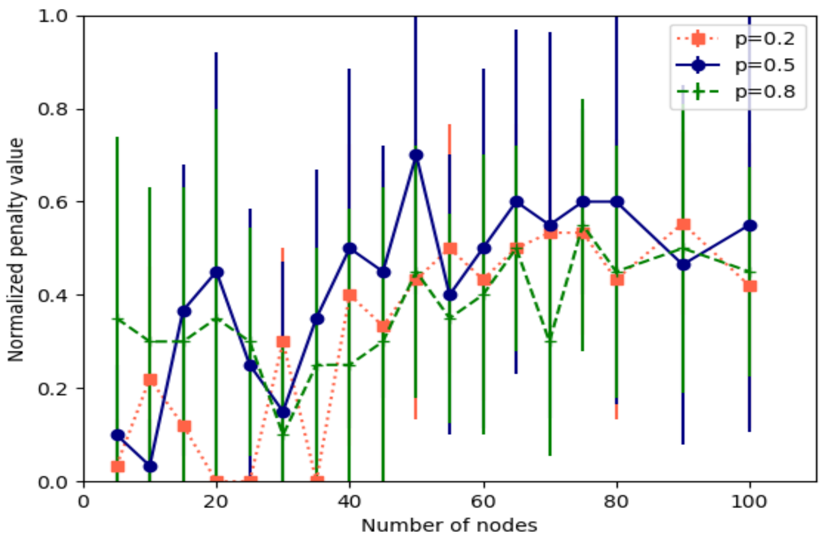


Figure 2.3: Penalty values of Method 3 w.r.t. Method 2

Number of nodes (N)	p'=0.2		p'=0.5		p'=0.8	
	Method 2	Method 3	Method 2	Method 3	Method 2	Method 3
10	11.038	0.007	12.266	0.007	15.430	0.009
20	39.889	0.046	55.927	0.063	66.355	0.075
30	103.791	0.176	139.259	0.230	173.447	0.282
40	200.488	0.445	297.024	0.642	363.224	0.775
50	387.377	1.039	517.641	1.362	680.101	1.791
60	589.751	1.898	889.882	2.689	1174.840	3.518
70	899.419	3.348	1354.019	4.922	1836.660	6.342
80	1247.091	5.244	1861.977	7.759	2488.567	10.279

Table 2.2: Mean time consumed (in seconds) by Method 2 and method 3

Observation Although we see that using Method 3 we get erroneous values while calculating the Shapley value of the nodes, but as this one is the fastest of the three methods and we use a very large dataset, that contains million of sessions, for our evaluation, we use Method 3 ([36]) to calculate Shapley values throughout the rest of the thesis.

Chapter 3

Proposed content placement framework

In this section we discuss in detail the proposed content placement framework and how it is evaluated in real life scenario. Most of the works of this chapter have already been published [9].

3.1 Problem formulation

3.1.1 Some basic definitions

In this section we elaborate with some examples, why game theoretic centrality is relevant for content placement in sparsely connected edge networks. Given a set of interconnected users, our problem aims at placing the content such that any user can access to its content with a minimum number of hops and maximum sharing. Each user has two ways of getting his/her wished content: either from the server using the fixed access network or from neighbors using the Wi-Fi network. In this problem, we assume that all households have Wi-Fi routers along with some storage units (like Apple TV or Chromecast). However, the Wi-Fi connectivity strongly depends on the transmitted power, the receiver sensitivity, and the path loss, which depends on the distance between transmitter and receiver. Hence, the Wi-Fi network has low connectivity and hence, a user may potentially require more than one hop to reach its/her content.

Apart from the user connectivity, the problem relies on the concept of *User Interest Probability (UIP)*, $p_{u,v}$, which is defined as the probability that a certain user u will watch a certain content v within a time period $(0, T]$. The higher is the UIP value for a content, the more is the chance that the user is actually going to access the content. This value can be calculated based on user's previous watching history, the video's popularity and many other factors. To calculate these probabilities, different machine learning techniques are available (e.g., [38]).

Given a certain content v, the user with the highest UIP is denoted as the most interested user of v and the user with the lowest UIP is denoted as the least interested user of v.

Definition 1 (Interested User) *Given a certain content v, the user with the highest user interest probability is denoted as the most interested user of content.*

For example, let us assume that for a particular video v_1, users u_1, u_2 and u_3 have a user interest probability of p_{u_1,v_1}, p_{u_2,v_1} and p_{u_3,v_1} respectively. If $p_{u_1,v_1} > p_{u_2,v_1} > p_{u_3,v_1}$, then we can say that u_1 is the most interested and u_3 is the least interested user in v_1. We can also say that user u_1 is more interested than u_2 and u_2 is more interested than u_3 in v_1. If $p_{u_1,v_1} = p_{u_2,v_1}$ then both the users u_1 and u_2 are equally interested in v_1.

3.1.2 Payoff definition

A content provider prefers to select those users as cache locations who - 1) is interested in the content, and 2) has a good sharing capability so that others can get the content from that user. This in turn helps in increasing traffic savings and reducing load on the server. Here we disregard the impact of storage and assume that each user always has enough storage capacity to cache a content. Later we will show that this is not always true in real cases and how much impact storage units have on performances.

Based on this UIP, we define the payoff function of our game. In our game, users are encouraged to cache content of their interest and/or content that can be shared with other nearby users. Hence, the payoff function of a coalition is equal to the sum of all $UIPs$ of the users of the coalition when they are able to share content among each other; otherwise, the payoff function is null. Mathematically, it can be expressed for a certain video v as

$$pf_v(S) = \begin{cases} \sum_{i \forall i \in S} p_{i,v} \text{ if } S \text{ is connected} \\ 0 \text{ otherwise} \end{cases} \tag{3.1}$$

where $p_{i,v}$ is calculated at the time of content placement only (i.e., at $t = 0$). It is to mention that, for different items, UIP values vary and so do the payoff values.

Based on the payoff function for each possible coalition, the Shapley value for each node can be calculated using Equation 3.1. The node with the largest Shapley value, will be selected to cache v.

Now our proposed caching strategy can be summarized as a three step process -

- *Step 1*: For a video v_1, get the user interest probability values for all the nodes in the network.
- *Step 2*: Calculate the payoff for each possible coalition using Equation 3.1.

- *Step 3*: Calculate the Shapley value for each node using Equation 2.6. The node with the highest Shapley value stores v_1.

Let us consider the example described in Section 2.3 and discuss two different content items v_1, and v_2:

Content item v_1

Let us consider the network of Fig. 2.1. Let us assume that for a particular video v_1, the *UIPs* are p_{A,v_1}=0.95, p_{B,v_1}=0.85, p_{C,v_1}=0.2, p_{D,v_1}=0.1, p_{E,v_1}=0.3 as shown in the figure. Based on these values, user A can be considered as the most interested on v_1 whereas C can be considered as the least interested user in v_1. In this case, for all the coalitions, the payoff function is calculated using the payoff definition of Eqn. 3.1. For example, $pf_{v1}(\{A\}) = 0,72$, $pf_{v1}(\{A, B\}) = 1,42$, $pf_{v1}(\{A, C\}) = 0$ (A and C are not a connected subgraph), $pf_{v1}(\{A, B, D\}) = 2,02$, etc. Based on these payoff functions, the Shapley values for each node are calculated, which are: $\zeta_A(pf) = 0.281$, $\zeta_B(pf) = 1.081$, $\zeta_C(pf) = -0.081$, $\zeta_D(pf) = 1.059$ and $\zeta_E(pf) = -0.035$. As B has the highest Shapley value, it is the selected location for v_1.

Content item v_2

Suppose for another video v_2, the values of the probabilities are p_{A,v_2}=0.1, p_{B,v_2}=0.09,p_{C,v_2}=0.9,p_{D,v_2}=0.35, p_{E,v_2}=0.29. Based on these values, user C can be considered as the most interested on v_1 whereas B can be considered as the least interested user in v_2. In this case, for all the coalitions, the payoff function is calculated just like above. Based on these payoff functions, the Shapley values for each node are calculated, which are: $\zeta_A(pf) = -0.178$, $\zeta_B(pf) = 0.471$, $\zeta_C(pf) = 0.991$, $\zeta_D(pf) = 0.344$ and $\zeta_E(pf) = 0.060$. As C has the highest Shapley value, it is the selected location for v_2.

Let us compare the proposed strategy with the state-of-the-art centrality measures. The state-of-the-art centrality measures do not take into account the user preferences for different videos and hence, for this particular example, both contents v_1 and v_2 will be cached at D. But as game theoretic centrality takes into account the user preferences, v_1 and v_2 are cached at B and C respectively (as shown in Fig. 2.1).

3.2 Proposed solution

In this Section we provide a complete framework for content placement and evaluation in a connected edge network. The framework consists of four sequential steps:

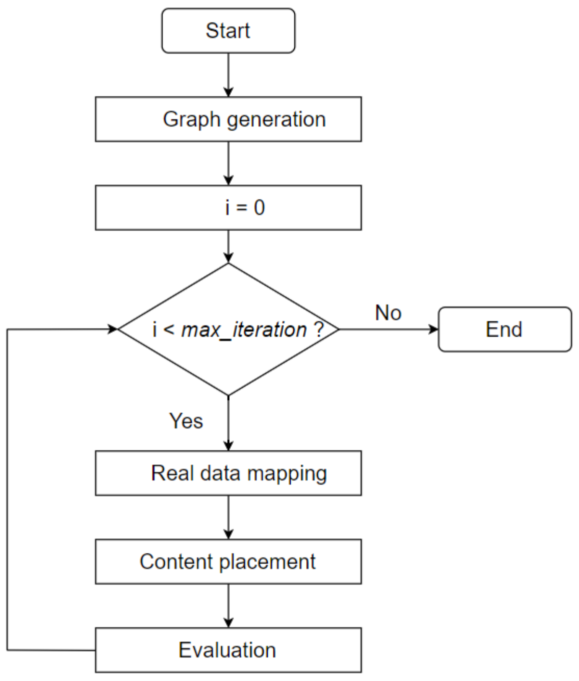


Figure 3.1: Flowchart for content placement and evaluation

1. **Graph generation** : In this step, we select the nodes for content placement and based on the pathloss model we consider, we generate a graph.
2. **Real data mapping** : In this step we introduce the dataset and map the real life data to the graph.
3. **Content placement** : In this step we calculate the Shapley values for each node and place content in the selected node(s).
4. **Evaluation** : In this step we formulate and measure all the parameters relevant to the simulation.

We perform steps 2 - 4 a predefined number of times which we call *max iteration*. In all the simulations, the value of *max iteration* is set at 20 unless stated otherwise. The steps are depicted in the flowchart shown in Fig. 3.1.

If the network we want to study is very large, then we consider splitting the users into small clusters into several clusters following a predetermined clustering algorithm. The necessity of doing the clustering as well as the detailed procedure are described in Section 3.2.5. However, as this step is not mandatory and applies only when the network is big, we do not keep it in the content placement and evaluation framework depicted in Fig. 3.1.

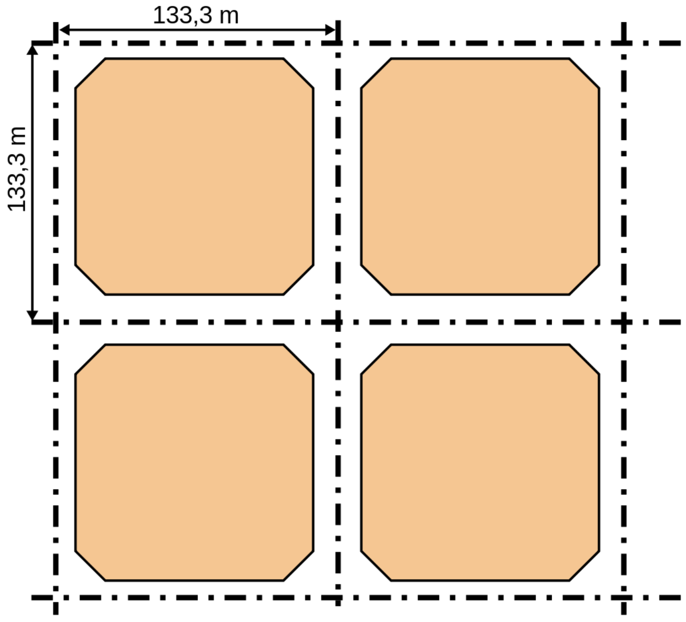


Figure 3.2: Example street and block layout [39]

3.2.1 Graph generation

We study the performance of our proposed strategy in a real life scenario. To do so, we consider an area consisting of long straight streets with strict grid patterns crossed by wide avenues and square blocks.

Fig. 3.2 shows the basic building distribution block which is found in large cities in the world like Barcelona, Manhattan, etc. Similarly we also consider an area with a long street on both sides of which square shaped buildings are situated. We assume that the street has a length of 133.3 meters [39] and each building has a length and width of 7 meters. Based on the real life scenario we consider, there can be single or multiple routers in each building and we calculate the link capacity [1] between every pair of routers following pathloss models provided by ITU (International Telecommunication Union). We consider two types of pathloss models:

1. **Outdoor propagation model** : This is the pathloss model between two routers are in two different buildings.
2. **Indoor propagation model** : This is the pathloss model between two routers are in the same building.

[1]The python code to calculate the link capacity between any two houses or any two apartments following the pathloss model described in this thesis is available here: http://tiny.cc/jft75y.

Parameter	Value
α	2.12 dB
β	29.2 dB
γ	2.11 dB
σ	5.06 dB
ξ	15 dB

Table 3.1: Values of parameters used in pathloss model

It is obvious that if we consider a scenario with multiple buildings and each building has multiple apartments, we need to use both the outdoor and the indoor pathloss model.

Outdoor pathloss model

Based on the distance between any two of the buildings and the number of buildings which are in between them, we build a very realistic outdoor path loss model based on ITU-R P.1411-9 [10]. We assume that Wi-Fi signal will operate at a frequency $f = 2.4$ GHz. Then the pathloss (PL) can be formulated as

$$PL_{outdoor} = 10 \times \alpha \times \log d + \beta + 10 \times \gamma \times \log f + N(0, \sigma) + (k+1) \times \xi \ dB \tag{3.2}$$

where d is the distance from source to destination building in meter, k is the number of buildings that lie in the LoS path between source and destination building and ξ is the building entry loss. For residential area, N is a zero mean gaussian random variable with a standard deviation σ. For a residential area, the values described in Table 3.1 have been used, which have been obtained from [10]. The value of building entry losses (ξ) can be found from ITU-R P.2109. The used value of ξ is valid for traditional houses only.

Indoor pathloss model

The indoor pathloss model is formulated based on ITU-R P.1238-7 [11] which provides propagation data and prediction method for the planning of indoor radiocommunication systems. The pathloss is given by

$$PL_{indoor} = 20 \times \log f + \beta + N_{pc} \times \log d + Lf(n_f) - 28 \ dB \tag{3.3}$$

where f is the frequency in MHz, N_{pc} is the distance power law coefficient, d is the distance between the sender and receiver in meter, $Lf(\cdot)$ is the floor penetration loss factor function and n_f is the number of floors between the sender and the receiver. If there are n_f number of floors between the sender and the receiver, then the total loss due to floor penetration is given by $Lf(n_f)$ dB. For apartments, the value of $Lf(\cdot)$ is assumed to be 10 and it is independent of the value of n_f. For residential area, the value of N_{pc} is given as 28 dB when f is 2.4 GHz.

3.2.2 Real data mapping

At the first part in this section we define the input parameters to generate a graph and in the next part, we elaborate the steps needed to start the evaluation. We also study how the value of each parameter varies when the value of storage and predictor accuracy varies. We assume that in the default case, each router has a speed of 400 Mbps and the transmitter power of antenna is 20 dBm. However, during simulations, we vary the router speed up to 1 Gbps and transmitter power up to 30 dBm. Assuming the noise power to be unity at the receiving end, we calculate the link capacity between each pair of routers. If the link capacity between any two routers is greater than the minimum bitrate required to play any certain video, only then the video can be shared between them. Throughout the simulations default value used for predictor accuracy is 0.8 and the default value used for storage is: 80%. We use the default values in all the simulations unless stated otherwise.

The results generated in this Section are based on the dataset described in Section 3. In the trace file, each line corresponds to a unique session that contains 16 different information fields. Among them the following are of most importance to us -

- *uid_mapping_id* : a unique user ID assigned to each user
- *ip_mapping_id* : a unique string assigned to each distinct IP address. One or more number of users can be connected to the same IP address.
- *asset_pid* : a unique string assigned to each video in the library
- *min_bitrate* : Minimum bitrate which is required to play the video.
- *avg_bitrate* : Average bitrate during the time the video was being watched. Multiplying this value with the duration of video gives us an approximate value of how much traffic was consumed during the whole watch.
- *request_time*: the date and time the content was requested.

If we consider n_b number of routers, from the trace file we randomly choose n_b IP addresses and randomly assign one of them to each router. For the sake of generality, we perform this step 50 times. From the trace file we also get the list of videos that has been requested by each of those IP addresses and the minimum bitrate required to play each video. We vary different parameters one at a time and find out the impact of the parameter on the performance of our proposed strategy.

3.2.3 Content placement

For each content, Shapley value is calculated for each node and based on the selected option (single or multiple placements), content is placed in the selected node(s).

3.2.4 Evaluation

At the first part in this section we describe the parameters we measure after the content placement is done, and at the second part we discuss the input parameters which are varied while we do the measurements.

Performance metrics

In this section we describe the parameters we measure after the content placement is done.

Traffic savings The traffic savings measurement states how much savings in traffic can be obtained when multiple users share the same content item. Let us denote as T_1 the traffic incurred while fetching the item from the server and as T_2 traffic incurred for each sharing (that is, from any user to the node where its content is placed). In general, $T_1 \gg T_2$. If m users fetch the item from server and n users share from them, then the traffic savings will be $1 - \frac{(m \times T_1 + n \times T_2)}{(m+n) \times T_1}$.

In our example of Fig. 2.1, if A, B and C watch v_1 and A and C get it from B, in that case the traffic savings would be 66.67%. This metric also tells us the factor by which the load on the server is reduced.

Average hopcount per sharing $\overline{h}$ This is a measurement of how closer our proposed strategy has been able to put the content near the interested user(s). It is also a measurement of latency in the network. The lower $\overline{h}$ is, the lesser number of hops is required by a node to get the wanted content, i.e., the closer the content has been put to the interested user. If user n_1 gets the content via h_1 hops, n_2 gets the content via h_2 hops, . . ., n_k gets the content via h_k hops, then $\overline{h} = \frac{h_1 + h_2 + \ldots + h_k}{k}$. Taking the example of Fig. 2.1, if A and C want to watch v_1, they can get it from B. A can get it via 1 hop while C can get it via 2 hops. So, in this case $\overline{h} = 1.5$.

Green users A green user is the one who only gets the content in less than or equal to one hop, i.e., who either caches the content or gets it directly from his/her neighbor via single hop. A green user does not utilize any link capacity from other users. The more is the number of green users, the better is the utilization of storage space and link capacity in the network. For example, if A wants to get a content v_2 from C, A has to get it via D and B. If none of B and D watch v_2, It means for a single sharing, three links - the links between $\{C, D\}$, $\{D, B\}$ and $\{B, A\}$, are being occupied which could have been used for more number of sharing.

Energy savings We evaluate potential energy savings from peer-assisted content delivery. We measure the savings by calculating how much energy, which otherwise would be consumed by delivering content from the original server, can be saved if the content is instead delivered from nearby users in the network (i.e., peers). We compute the consumption of energy in both cases as per-bit energy cost functions. $\Psi_s(.)$ denotes the energy cost function when the content is retrieved directly from the server and $\Psi_p(.)$ denotes the energy cost function when the content is retrieved from nearest peer. If a user consumes $\mathcal{B}$ bytes of a content, the amount of energy needed to get it directly from the content server is $\Psi_s(\mathcal{B})$ and the amount of energy needed to get it from a peer is $\Psi_p(\mathcal{B})$. Thus, the amount of savings in energy is

$$S_E = 1 - \frac{\Psi_p(\mathcal{B})}{\Psi_s(\mathcal{B})}$$

The energy model we use provides *per-bit* and *per-hop* energy consumption values based on actual measurements [40]. In this model, consumed energy is proportional to the number of bytes consumed ($\mathcal{B}$). In other words, $\Psi_s(\mathcal{B}) = \mathcal{B} \times \psi_s$ and $\Psi_p(\mathcal{B}) = \mathcal{B} \times \psi_p$, where ψ_s and ψ_p are the two proportionality constants for the two different cases.

Per-bit energy cost for delivering from servers (ψ_s)

Following the Valancius model [40],

$$\psi_s = PUE(\gamma_s + h_{dc} \times \gamma_r) \tag{3.4}$$

where PUE is the Power Usage Efficiency metric (defined as the ratio between total power consumed by a data center and the power actually delivered to its IT equipment), γ_s is the server cost, γ_r is the receiver cost and h_{dc} is the number of hops necessary to reach the client in a conventional, centralized system [40].

Per-bit energy cost for delivering from peers (ψ_p)

Following the same model,

$$\psi_p = 2 \times l \times \gamma_n + PUE \times h_n \times \gamma_r \tag{3.5}$$

where l is the energy loss factor in end-user equipment, γ_n is the user cost and h_n is the number of hops required to reach the client. The factor 2 accounts for simultaneous downloading and uploading the contents with other peers. The values of the parameters we use in the simulations are given in Table 3.2.

Input parameters

In this section we discuss the input parameters which are varied while we do the measurements. The four performance metrics have been measured with respect to the following parameters:

Parameter	Value
PUE	1.2
γ_s	211 J/Gb
h_{dc}	14.01
γ_r	150 J/Gb
l	1.07
γ_n	100 J/Gb

Table 3.2: Values of parameters used in Valancius energy model [40]

Storage is defined as the combined storage of all the users. If during a certain time interval, all these users watched n number of different videos in total, then 10% storage means all of their caches combined can store $0.1n$ videos.

Predictor accuracy indicates how correctly the user preference is guessed. The higher the predictor accuracy is, the more accurately it can predict if a user is going to watch a content.

Cache eviction policy describes how to replace an existing content in the cache when a new content arrives. We use following four cache replacement policies -

- Least Recently Used (LRU) - the most well-known and most widely used replacement policy. When a new content needs to be inserted into the cache, the least recently requested one gets evicted.
- First In First Out (FIFO) - When a new content is inserted, the evicted content is the first one inserted into the cache.
- Last In First Out (LIFO) - When a new content is inserted, the evicted content is the last one inserted into the cache.
- Random (RAND) - When a new content is inserted, the evicted content is chosen randomly from the contents existing in the cache.

In all our simulations, we use FIFO as the default cache eviction policy unless stated otherwise.

In real life, if the number of nodes in a graph becomes very high, sometimes the number of hops between two nodes sharing content can become more than the number of hops between the server and the requester node. This is the reason we do all our simulations in a k hop restricted network, where $k \in [1, \infty)$. Based on the value of k, we consider four types of hop restricted network:

1. **One hop restricted network** : In this type of network, a node can get a content only from its immediate neighbors.

2. **Two hop restricted network** : In this type of network, a node can get a content from all the nodes which are within 2 hops away from it.

3. **Three hop restricted network** : In this type of network, a node can get a content from all the nodes which are within 3 hops away from it.

4. **Unrestricted network** : This is a hypothetical scenario where there is no restrictions on hops between any two content sharing nodes. *It is to be noted that in a hop unrestricted network, traffic savings always remains same no matter whichever node the content is placed in.*

In the next chapter we evaluate our proposed strategy in different real life scenarios by measuring the parameters described above.

3.2.5 User clustering

This step is not mandatory in the content placement and evaluation framework. This step is performed when necessary after Step 1 (Graph generation). In this Section we focus on clustering the users into smaller groups for the reasons described below:

- We already showed that the calculation of Shapley value has a very high time and space complexity when the number of nodes gets high. Although using Method 3 described in Section 2.4 reduces the computation time, from Fig. 2.3 we see that the result differs from the real Shapley values.
- In a very large network (like, a network formed in a city with more than 10,000 users) placing just one content is not enough. We can cluster the users into several groups and in each cluster we can place a content for better resource utilization.

We consider splitting the users into small clusters (C_n) of size n each. There are two things to be concerned with while determining the size of the cluster:

- When the number of users in a network decreases, user diversity decreases, the probability that two or more number of users are sharing the same content also decreases which in turn reduces the traffic savings.
- When the number of users in a network increases, content sharing also increases but the time and space complexity of the Shapley value computation also increases.

In this section we try to determine the value of the cluster size n in a way so that balance can be obtained between these factors by calculating two types of probability values.

Probability of content sharing (p_{cs})

This is the probability that a content will be shared by two or more number of users. This probability value is calculated on a per content basis. Let us assume there are 3 users

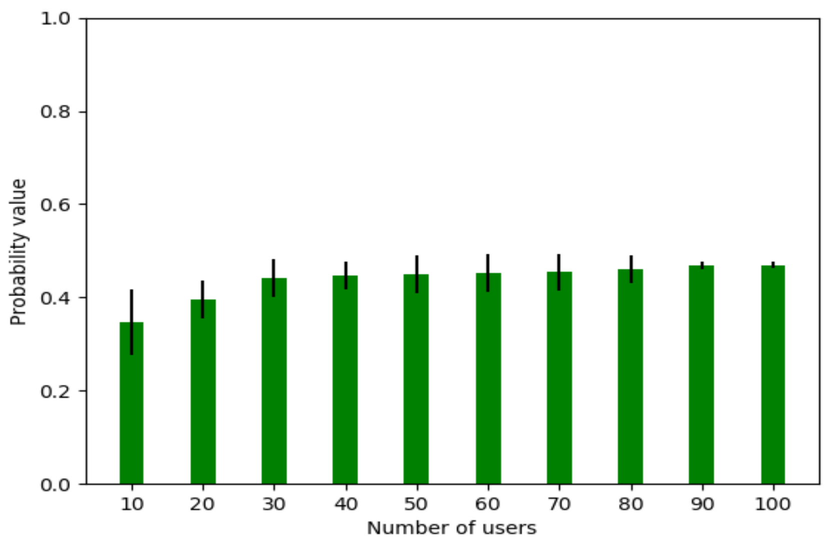
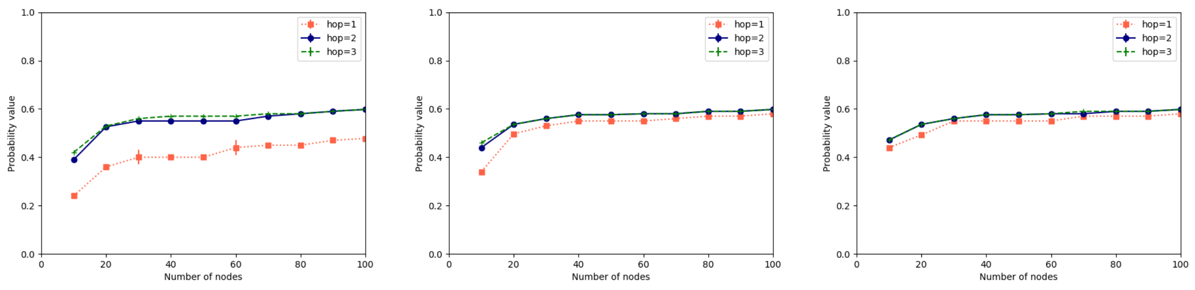


Figure 3.3: Probability of content sharing (p_{cs}) with cluster size

u_1, u_2 and u_3 who share content among them. User u_1 watches content c_1, c_2, c_3, user u_2 watches content c_2, c_3, c_4 and user u_3 watches c_4, c_5. If we combine their watching results together, we can see that there are 5 unique contents $\{c_1, c_2, c_3, c_4$ and $c_5\}$. We call this set as content universe. Among them, c_2, c_3 and c_4 have been watched by 2 or more different users. So, the probability of content sharing of this content universe in this case (p_{cs}) is $\frac{3}{5}$. To calculate the p_{cs} under different scenarios, we do the following simulations.

Simulation 1 At first we form a cluster C_n of size n by selecting n number of users randomly from the BBC iPlayer dataset, where n can vary between 10 and 100. We run this simulation 20 times, each time selecting the users randomly from the dataset and plot the results in Fig. 3.3 with error bars. From Fig. 3.3 we see that when the number of users in a cluster increases, the chance of another user watching the same content also increases. We also observe that the p_{cs} value becomes almost constant after a certain point because with increased number of users content sharing increases, but at the same time the size of the content universe also increases and p_{cs} value becomes almost constant.

Simulation 2 In this section also we calculate the value of p_{cs} in a network under different circumstances. We generate graphs according to Erdos-Renyi model [37] for various (n, p) values where n is the number of users in the graph and p is the probability that an edge exists between a pair of nodes. We randomly select a node in the graph and then form a cluster by selecting all its neighbors which are i number of hops away from it. From the BBC iPlayer dataset we randomly select n number of IP addresses, assign them randomly to each of the nodes and calculate the p_{cs} value in the cluster. We vary p between 0.2



(a) p=0.2 (b) p=0.5 (c) p=0.8

Figure 3.4: Probability value of content sharing between multiple users

and 0.8, where $p = 0.2$ implies a sparsely connected graph and $p = 0.8$ implies a densely connected graph, n between 10-100 and i between 1-3.

In general, when the number of hops increases, the number of neighbors also increases, and thus, the probability that a content will be shared by 2 or more number of users also increases. When the graph is sparsely connected ($p = 0.2$), in a single hop the value of p_{cs} varies between 0.2-0.4 and when the number of hops is 2 or 3, the probability value lies between 0.4-0.6. For densely connected graphs ($p = 0.5, p = 0.8$), for all number of hops, the value of p_{cs} varies between 0.4-0.6.

Reusability probability (p_{ru})

This probability value tells us how much a content will be used by other users once it is accessed by a particular user for the first time. This value gives us an impression of how many different users access the same content within a certain time interval. p_{ru} value is calculated on a per session basis. Suppose there are 4 users u_1, u_2, u_3 and u_4 who share content among them. During a certain time period $[t_1, t_8]$, their combined session history looks something like -

Time	User	Content ID
t_1	u_1	c_1
t_2	u_2	c_2
t_3	u_3	c_1
t_4	u_4	c_1
t_5	u_2	c_2
t_6	u_4	c_3
t_7	u_3	c_3
t_8	u_1	c_5

Table 3.3: Example - session records

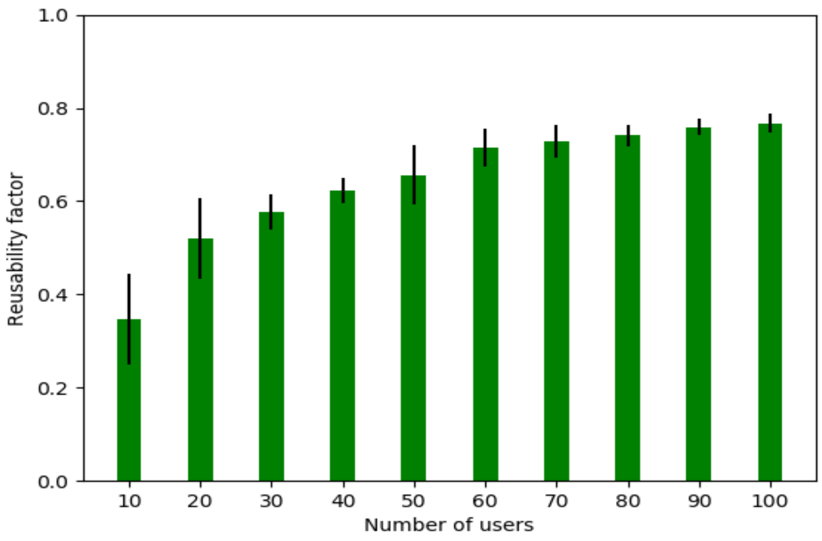


Figure 3.5: Reusability probability with cluster size

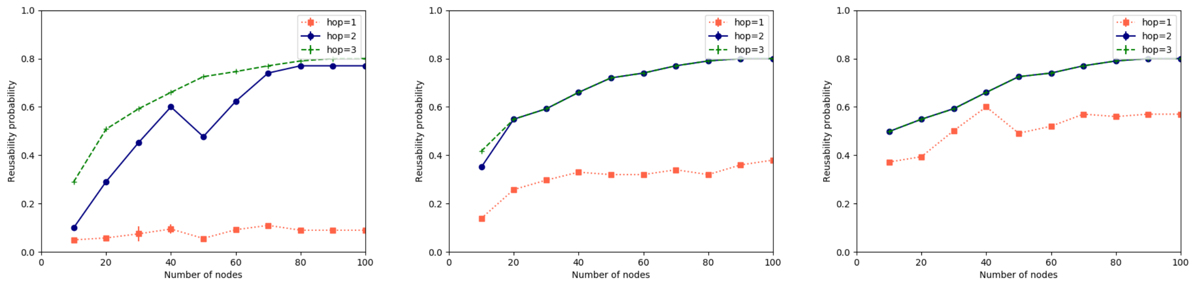
From Table 3.3 we see that c_1 has been accessed by u_1 at first and later has been accessed again by u_3 and u_4, i.e. it has been re-accessed 2 times. Similarly c_2 has been re-accessed 1 time by u_2 at t_5 after u_2 first watched it at t_2 and c_3 has been re-accessed 1 time by u_3 at t_7 after u_4 first accessed it at t_6. So, in this example, the probability that a content will be re-accessed after it has been accessed for the first time is $\frac{4}{8}$. The higher the value is, the more is the traffic savings.

Simulation 1 Here also we run the simulation exactly the same way described in Simulation 1 in previous section.

We can see from Fig. 3.5 that this value increases gradually with the number of users.

Simulation 2 Here also we run the simulation exactly the same way described in Simulation 2 in previous section.

Between two probability values we discussed (p_{cs} and p_{ru}), p_{ru} does not converge when the number of nodes are within 100, but p_{cu} value becomes almost constant when the number of nodes are between 30 and 100. So, when the size of cluster is 30, the content sharing is better than the cases when the cluster size is 10 or 20 and the content sharing is same when the cluster size is 100. But when the cluster size is 30, Shapley value computation will be much faster than with cluster size of 100. That is why we choose 30 (or, some value close to it) as the ideal cluster size for this kind of content sharing scenario.



(a) p=0.2 (b) p=0.5 (c) p=0.8

Figure 3.6: Reusability probability

Clustering algorithms

There are several existing graph partitioning algorithms ([41,42]) which partition a graph into several subgraphs. We also partition a graph into several subgraphs and consider each subgraph to serve as a cluster. In the context of content sharing, we can partition a graph two ways -

1. *Modified Kernighan-Lin algorithm* - we modify the Kernighan-Lin algorithm [41] so that we can partition the graph into several subgraphs such that maximum number of users in a subgraph remains less than or equal to a fixed number. Using this algorithm, we can partition the graph into subgraphs of specific size, so that we can control the number of copies of a content in the network and this way we can control the traffic savings.

2. *Greedy graph partitioning algorithm* - we propose this new graph partitioning algorithm. Using this algorithm we can partition a graph into subgraphs in such a way so that we can control the maximum number of users as well as maximum number of hops between any nodes in a subgraph. This way we can control both the traffic savings and the latency.

Below we discuss both the algorithms in detail.

Modified Kernighan-Lin algorithm The Kernighan-Lin algorithm partitions an undirected graph $G = (N, E)$ into two subgraphs of equal or nearly equal size. The goal of the algorithm is to partition N into two subsets X and Y of equal or nearly equal size and to minimize the number of crossing edges from X to Y. For each $x \in X$, let I_x be the sum of weights of edges between x and other nodes in X and E_x be the sum of weights of edges between x and nodes of Y. Similary, let I_y be the sum of weights of edges between y and other nodes in Y and E_y be the sum of weights of edges between y and nodes of X. Also, for any node s, let us define the difference between E_s and I_s as $D_s = E_s - I_s$. If x and

y are interchanged, then the reduction in weight is

$$T_{old} - T_{new} = D_x + D_y - 2 \times w_{x,y}$$

where $w_{x,y}$ is the weight of possible edge between x and y. The algorithm tries to calculate an optimal series of interchange operations between elements of X and Y such that $T_{old} - T_{new}$ is maximized. Given a graph with n number of vertices, each pass of the algorithm runs in time $\mathcal{O}(n^2 \log n)$.

As we already discussed earlier, in our case we want to keep the maximum number of users in a cluster close to 30. To do so, given a graph, we iteratively partition it into smaller subgraphs until the number of nodes in each cluster satisfies the condition. Our proposed algorithm is described in detail in Algorithm 1.

Algorithm 1 Modified Kernighan-Lin algorithm

```
 1: Input: Original graph -$g$, maximum cluster size - $s$
 2: Output: List of subgraphs with size of each $\leq s$
 3: all subgraph list $\leftarrow$ list of all subgraphs
 4: max subgraph $\leftarrow$ subgraph with highest number of nodes from all subgraph list
 5: Add $g$ to all subgraph list
 6: while number of nodes in max subgraph $\geq$ maximum cluster size do
 7:     partition $g$ into two subgraphs $g1$ and $g2$
 8:     add $g1$ and $g2$ to all subgraph list
 9:     select max subgraph from all subgraph list
10: end while
```

Greedy graph partitioning algorithm [2] The goal of this algorithm is to partition an undirected graph $G = (N, E)$ into a number of subgraphs in such a way that

1. The number of hops between any two nodes in the subgraph is less than or equal to a pre-defined *max hop* value.
2. The number of nodes in each subgraph is less than or equal to a pre-defined *maximum cluster size.*

We select a *start_node* from which we traverse the graph. The *start_node* can either be selected randomly or can be uniquely specified. We select the node with lowest degree in the graph to be the *start_node*. From there we select all the nodes which are directly connected to the *start_node*. If all the selected nodes are within a *max hop* value of one another, they are added to the same cluster, otherwise a new cluster is formed. Then we take the neighbors one by one and start traversing to their neighbors until all the nodes in the graph are visited.

[2]The python code for this greedy graph partitioning algorithm is available here: http://tiny.cc/9gz95y.

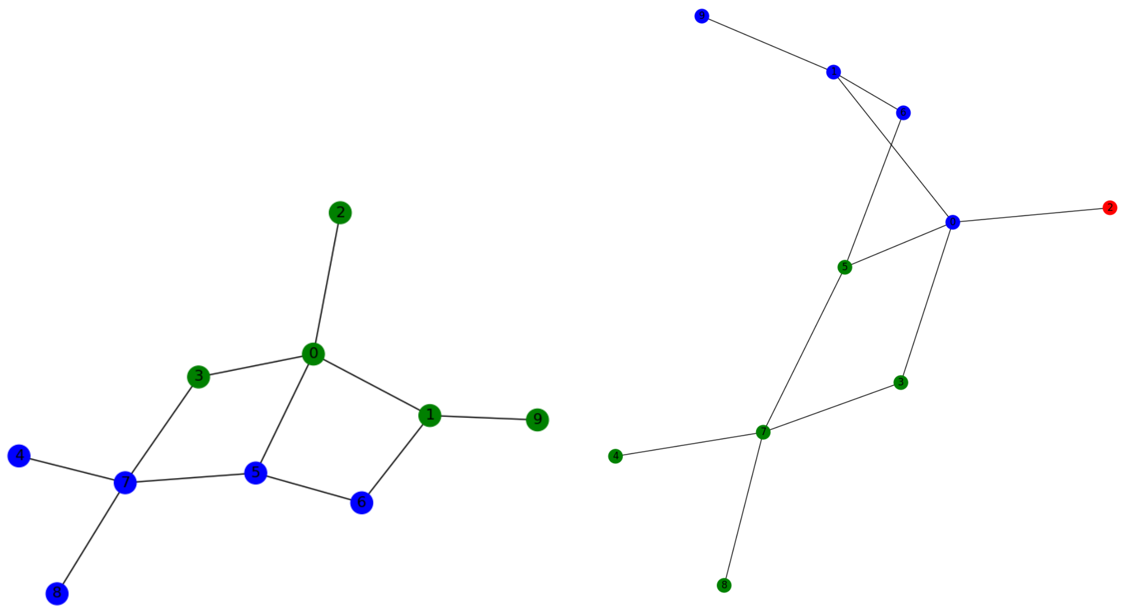
While a new node is visited, for all the existing clusters the two conditions mentioned above are checked. If a cluster is found where the above conditions are held, the node is added to that cluster. If multiple clusters are found, then the node is added to the one where the number of nodes is maximum. If no suitable cluster is found, then a new cluster is created and the node is added there.

As we can see, while traversing each node we always check the two aforementioned criteria, in the clusters generated all the nodes are within *max hop* away from one another. The only difference of this proposed algorithm with kernighan-Lin algorithm is that we cannot determine the number of clusters or number of users in each cluster beforehand. Our proposed algorithm is described in detail.

Algorithm 2 Greedy graph partitioning algorithm

```
1: Input: Original graph -$g$, maximum cluster size - $s$, maximum number of hops - $k$
2: Output: List of subgraphs with size of each $\leq s$
3: $unvisited\ nodes \leftarrow$ list of nodes in the graph
4: generate sequence following which nodes will be selected - $node\_sequence$
5: $count = 0$
6: while number of $unvisited\ nodes \geq 0$ do
7:     $node = node\_sequence[count]$
8:     $all\_clusters \leftarrow$ list of all available clusters for $node$
9:     for cluster in existing clusters do
10:        if $node$ is within $k$ hops away from all other nodes in the cluster then
11:            Add $cluster$ to $all\_clusters$
12:        end if
13:    end for
14:    if $all\_clusters$ is $\emptyset$ then
15:        create a new cluster and add $node$ to it
16:    else
17:        sort clusters in $all\_clusters$ in descending order of size
18:        for $cluster$ in $all\_clusters$ do
19:            if size of $cluster \leq s$ then
20:                Add $node$ to the $cluster$
21:                break
22:            end if
23:        end for
24:    end if
25:    $count + 1 \leftarrow count$
26:    Remove $node$ from $node\_sequence$
27: end while
```

From Fig. 3.7a we see how the partitions look like while using these two algorithms. As we can see, using Algorithm 3.2.5 we can have control over both the traffic savings and



(a) Partitioning using Kernighan-Lin algorithm (b) Partitioning using Greedy graph partitioning algorithm

Figure 3.7: graph partitioning algorithms

average hopcount value, we use this algorithm for user clustering whenever the need arises.

Chapter 4

Evaluation

In this section at first we show the advantages of content placement using game theoretic centrality over state-of-the-art centrality measures. We show that our proposed strategy puts the content closer to the user(s) who have a high chance of watching them and thus increases the energy savings and reduces latency. Then we study how our proposed strategy performs in two different real life scenarios.

4.1 Comparison of game theoretic centrality with other centralities

Let us now consider a more generic scenario of N users, who are connected with Wi-Fi and can share content among them. A user n_i has a Wi-Fi signal of bandwidth b_i of its own using which it can share content with other nodes and we assume $b_1 = b_2 = ... = b_N = b$. b_{ij} denotes the available bandwidth between the users i and j for sharing content. The minimum bitrate required to play a video is denoted by k. Both b_{ij} and k are normalized with respect to b.

For our simulations, we generate a sparsely connected graph with $N = 10$. We randomly assign a value between 0 and 1 to each of $b_{ij} \forall i, j \in \{N\}$ which signifies the link capacity. This value is set before the start of the simulation and does not change throughout the simulation. Now, for each content c, we assign another randomly chosen value between 0 and 1 that signifies the bitrate k of c. For all $i, j \in \{N\}$, if the value of b_{ij} is greater than k, then we assume that c can be shared between them and an edge is created between i and j. Also, for each user i, we randomly assign user interest probability values $p_{i,c} \forall i \in \{N\}$ and do a weighted random selection to decide which user(s) will watch c.

We run the simulation for 20 times, each time randomly assigning values to k and $p_{i,c} \forall i \in \{N\}$. We measure the traffic savings, average hopcount per sharing and green users with

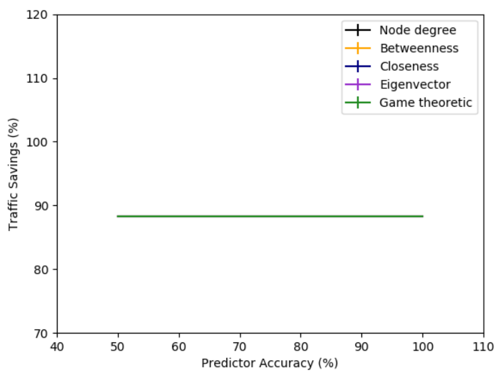


Figure 4.1: Traffic savings

respect to predictor accuracy. As this is not a real life dataset, we do not know precisely the size of the content universe and thus we do not perform measurements with respect to storage. For the same reason, we do not perform measurements on energy savings also. After we run all the experiments, then the average is taken and error bar is also plotted. We consider some of the existing centrality measures to find the most influential user in the cluster where the content can be cached and compare the performance with our proposed strategy.

4.1.1 Traffic savings

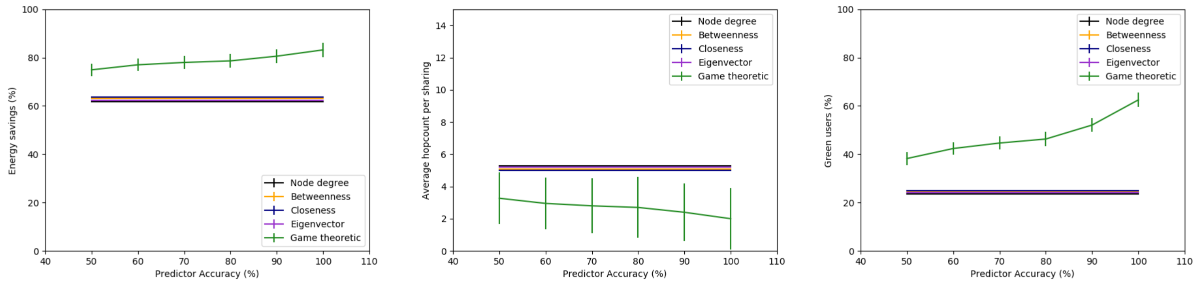
We measure the traffic savings while varying the predictor accuracy. From Fig. 4.1 we see that traffic savings are independent of the predictor accuracy and has the same value for all centrality measures. This is quite expected as we have already explained, in a hop unrestricted network, traffic savings is independent of the content placement node.

4.1.2 Average hopcount per sharing

The lower is the value of the average hopcount per sharing, the closer the content has been put to the interested user(s). From Fig. 4.2b we see that for our proposed strategy, this value is much smaller than the other ones. Following our strategy, a user is able to find content within 2-3 hops.

4.1.3 Green users

The more is the number of green users, the better is the resource utilization in the network. From Fig. 4.2c we see that for other centrality measures the number of green users is around



(a) Energy savings (b) Average hopcount per sharing $\overline{h}$ (c) Green users

Figure 4.2: Performance for varying parameters

25%, while using our strategy, the number of green users lies between 40-70%.

4.1.4 Energy savings

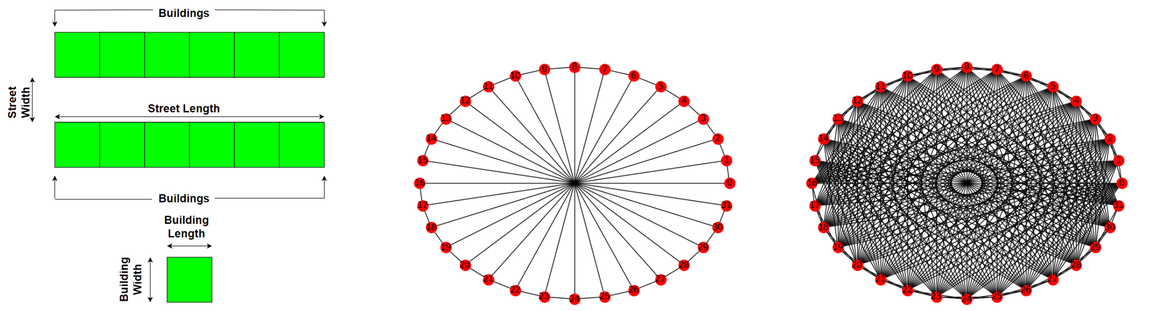
In Fig. 4.2a we plot the savings in energy obtained while we vary the predictor accuracy. When the predictor accuracy increases, energy savings increases for game theoretic centrality and remains constant for others. For game theoretic centrality, energy savings vary between 75-85% while for others this value lies between 60-65%.

4.2 Real life scenario 1 : Singlehouse scenario

For the first real life scenario, we consider a building distribution as depicted in Fig. 4.3a. Street length, street width and building width value is same as described in Section 3.2.1.

4.2.1 Scenario Selection & Network Formation

In this scenario, we get the coordinates of the center of the houses and calculate the distance between any two houses and number of houses situated on LoS path between any two houses using cartesian geometry. In the locality we choose, we assume all the buildings are single houses and there is only one router which is located at the center of each house. In reality, the router can be anywhere inside the house, but our assumption yields an error which can be of a few meters at maximum. Considering outdoor propagation model (as described in Section 3.2.1) as the pathloss model between the buildings, we calculate the link capacity between the buildings available for content sharing. In Fig. 4.4 we depict how the average node degree of the graph changes when the parameters are varied by 50%. We see that when transmitter power is reduced by 50%, the graph becomes disconnected and the average node degree becomes 0. When the building width is varied by 50%, the average



(a) Topview of building distribution (b) Graph 1 - Router speed = 400 Mbps, Tx power = 20 dBm (c) Graph 2 - Router speed = 1 Gbps, Tx power = 30 dBm

Figure 4.3: Generated graphs in singlehouse scenario

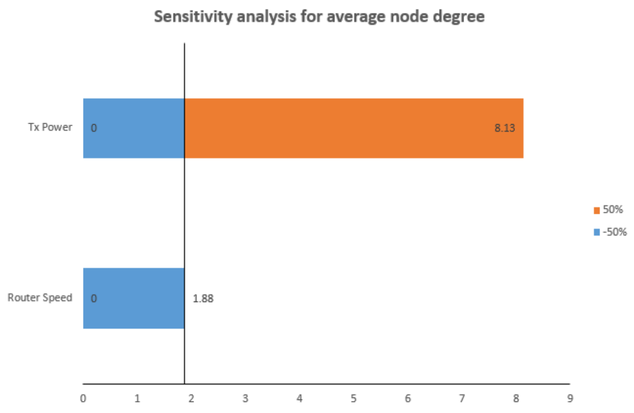


Figure 4.4: Tornado charts for average node degree in singlehouse scenario

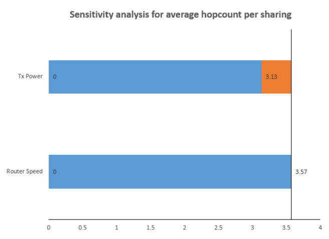
node degree changes slightly but when the router speed is varied, it remains constant. From Fig. 4.5 we get an idea of the importance of the parameters in generating the graph.

4.2.2 Real data mapping

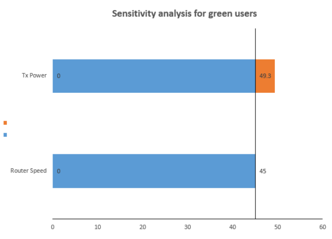
This simulation set up is exactly same as described in Section 3.2.2.

4.2.3 Content placement

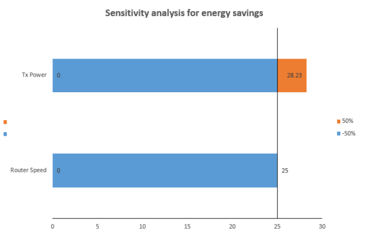
For each content, we calculate the Shapley value for all nodes in the graph and select the node with the highest Shapley value for content placement.



(a) average hopcount per sharing



(b) green users



(c) energy savings

Figure 4.5: Tornado diagrams for singlehouse scenario

4.2.4 Measurement of performance

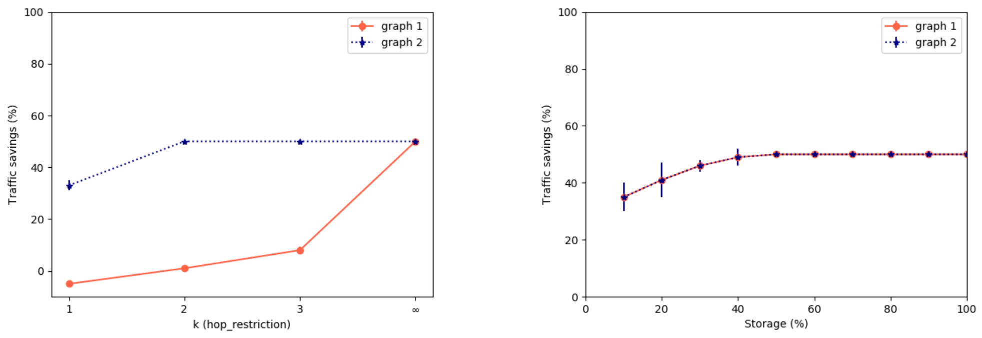
In this section we compare the performance of the proposed strategy in the scenario mentioned above. We assume in each house there is a 802.11ac router with a very high speed. From Fig. 4.5 we see that when one parameter is varied while the others are kept constant, the performance does not change much. That is why we consider two situations:

- **Default scenario** : In this scenario, the router speed and transmitter power are set at default values, i.e., router speed = 400 Mbps, transmitter power = 20 dBm.
- **Best case scenario** : In this scenario, both the parameters are set at their maximum possible values, i.e., router speed = 1 Gbps, transmitter power = 30 dBm.

Fig. 4.3b and Fig. 4.3c are two snapshots of the network in these two different situations. From this point onwards, we denote the graph generated in default scenario (router speed = 400 Mbps, transmitter power = 20 dBm) as Graph 1 and the graph generated in the best case scenario (router speed = 1 Gbps, transmitter power = 30 dBm) as Graph 2. In both the graphs, an edge between two houses exists only if the link capacity between the houses is greater than 700 kbps.

Traffic savings

We measure traffic savings in hop restricted and hop unrestricted networks with respect to all the parameters, i.e., storage, predictor accuracy and cache eviction policy and study the impact of each of these parameters on traffic savings. From Fig. 4.6a we see that in case of Graph 1, traffic savings increases when the value of k increases and the value becomes maximum when there is no hop restriction. This is quite understandable because when the value of k increases, more content sharing becomes possible and hence traffic savings increases. In case of Graph 2, traffic savings is always higher than that of Graph 1 and becomes constant for $k \geq 2$. This is because as the average node degree is much higher in Graph 2, for a particular k value more nodes are available for sharing content than in Graph 1. As we already pointed our earlier, in hop unrestricted case, the traffic savings is



(a) Different hop restricted networks

(b) Varying storage capacity in hop unrestricted network

Figure 4.6: Traffic savings in singlehouse scenario

same for both Graph 1 and Graph 2. It is to be noted that traffic savings is independent of predictor accuracy and cache eviction policy.

In Fig. 4.6b we study the impact of storage in traffic savings in a hop unrestricted network. We see that with 40% storage maximum possible traffic savings can be obtained for both Graph 1 and Graph 2.

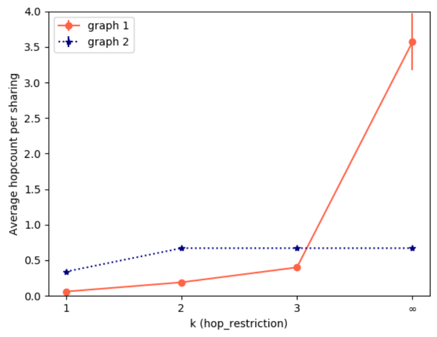
Average hopcount per sharing

We evaluate the average number of hops per sharing to study how close our proposed strategy has been able to put content to the interested user(s). As we can see from Fig. 4.7a, for $k \leq 3$, the average hopcount per sharing value is much lesser in Graph 1 than in Graph 2. This is because in case of a hop restricted network, in Graph 1 lesser content sharing takes place than in Graph 2. However, when the network becomes hop unrestricted, this value is much higher in case of Graph 1 than in case of Graph 2 because in Graph 2 average node degree value is higher so the average number of hops between any pair of nodes is lower.

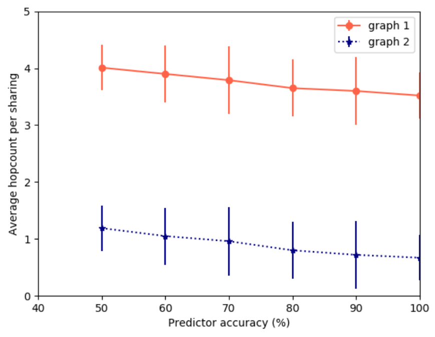
In Fig. 4.7b we study the impact of predictor accuracy in a hop unrestricted network. In both Graph 1 and Graph 2, the average hopcount value decreases as the predictor accuracy increases.

Green users

We evaluate the percentage of green users who get the content within 1 hop at maximum. We can see from Fig. 4.8a that in Graph 1, the percentage of green users decreases when the value of k increases and in Graph 2 this percentage value becomes constant for $k \geq 2$.

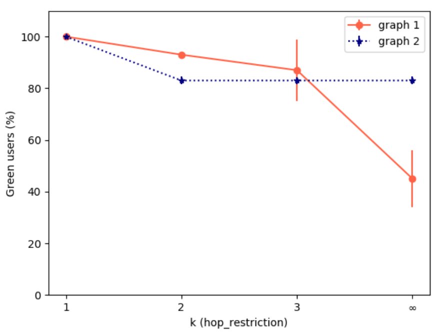


(a) Different hop restricted networks

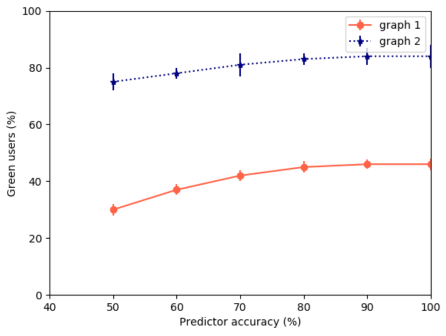


(b) Varying predictor accuracy in hop unrestricted network

Figure 4.7: Average hopcount per sharing in singlehouse scenario

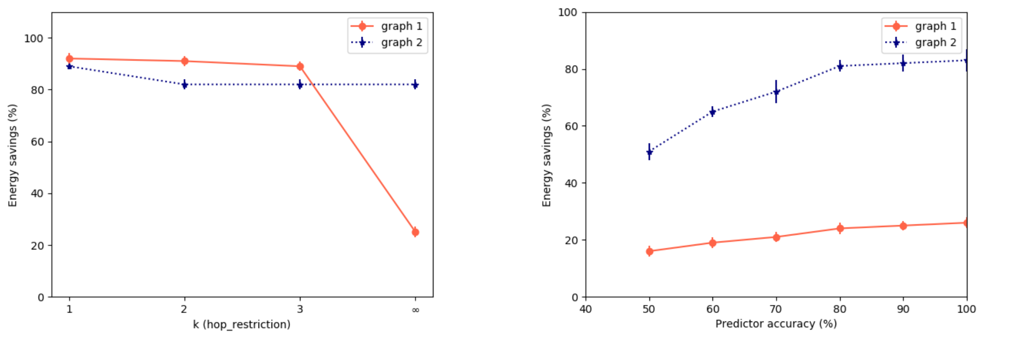


(a) Different hop restricted networks



(b) Varying predictor accuracy in hop unrestricted network

Figure 4.8: Green users in singlehouse scenario



(a) Different hop restricted networks (b) Varying predictor accuracy in hop unrestricted network

Figure 4.9: Energy savings in singlehouse scenario

From Fig. 4.8b we see that in an unrestricted network, when the value of predictor accuracy increases, for both Graph 1 and Graph 2 the percentage of green users also increases. This is because when the predictor accuracy increases, content are placed more closer to the interested users and more users get the content within a single hop.

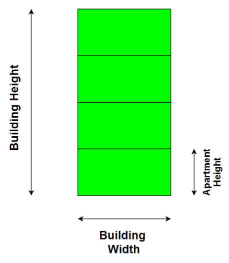
Energy savings

We measure energy savings with respect to all the parameters, i.e., storage, predictor accuracy and cache eviction policy and study the impact of each of these parameters on energy savings. As we can see from Fig. 4.9a, in Graph 1 energy savings drops sharply when the network becomes hop unrestricted because the average hopcount per sharing value increases. In Graph 2 the energy savings becomes constant for $k \geq 2$ because the average hopcount per sharing value also becomes constant in Graph 2 for $k \geq 2$.

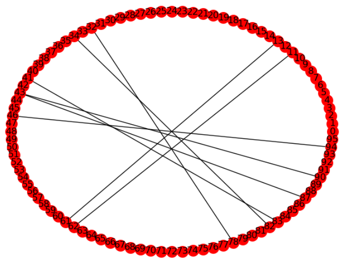
In case of a hop unrestricted network, energy savings increases when the predictor accuracy increases both in Graph 1 and Graph 2. This is quite expected because when the predictor accuracy increases, average hopcount per sharing value decreases which in turn increases the energy savings.

4.3 Real life scenario 2 : Apartment scenario

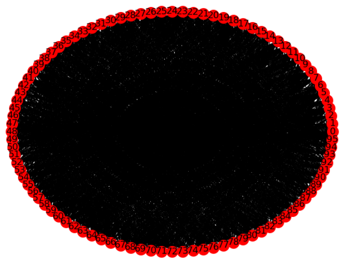
In Section 4.2 we consider a dense urban scenario, i.e., each building is a house itself and there is only one router in each building. The building distribution model is the same as described in Section 4.2. In this Section we consider that in each building there are multiple apartments and thus there are multiple routers in each building. For simplicity, we consider that in each building there are f_l number of floors and in each floor there is one



(a) Sideview of a building



(b) Graph 1 - Router speed = 400 Mbps, Tx power = 20 dBm



(c) Graph 2 - Router speed = 1 Gbps, Tx power = 30 dBm

Figure 4.10: Apartment Scenario

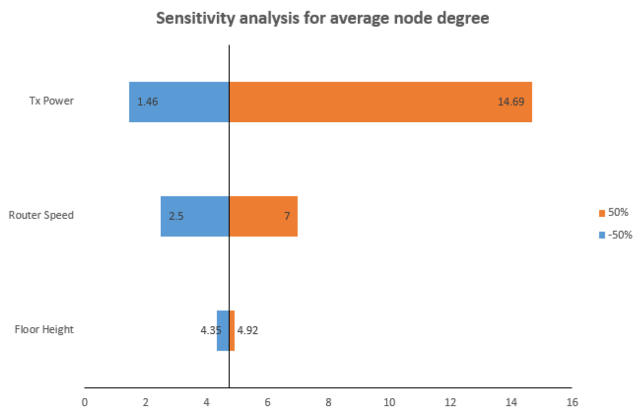


Figure 4.11: Tornado charts for average node degree in apartment scenario

apartment. In our model, we consider f_l has a value of 3 and the height of each apartment is 3.5 meters.

4.3.1 Network formation

We assume that in each apartment there is a router which is situated at the middle of the floor. In reality, the router can be anywhere in the apartment but our assumption yields an error of a few meters which does not have any significant impact on the evaluation results. Between the apartments on different buildings we consider the outdoor propagation model and between the apartments on the same building we consider indoor propagation model (Both of these models have been discussed in Section 3.2.1). In Fig. 4.11 we depict how the average node degree of the graph changes when the parameters are varied by 50%. We see that transmitter power plays the most important role and height of the floor plays the least important role on the average node degree in the generated graph.

4.3.2 Real data mapping

The simulation setup is same as described in Section 4.2.2.

4.3.3 Content placement

The content placement is done the same way as described in Section 4.2.3.

4.3.4 Measurement of performance

The measurement of performance is same as described in Section 4.2.4. We use the same set of values as described in Section 4.2.4 and generate two different graphs - Graph 1 and Graph 2 for evaluation purpose.

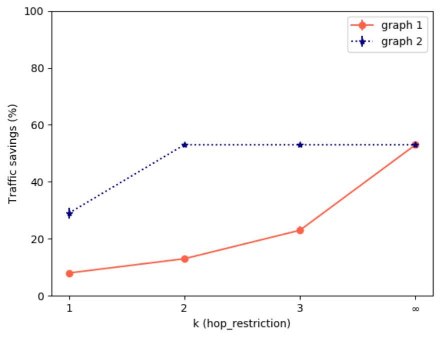
Traffic savings

We measure traffic savings in hop restricted and hop unrestricted networks with respect to all the parameters, i.e., storage, predictor accuracy and cache eviction policy and study the impact of each of these parameters on traffic savings. From Fig. 4.12a we see that in case of Graph 1, traffic savings increases when the value of k increases and the value becomes maximum when there is no hop restriction. This is quite understandable because when the value of k increases, more content sharing becomes possible and hence traffic savings increases. In case of Graph 2, traffic savings is always higher than that of Graph 1 and becomes constant for $k \geq 2$. This is because as the average node degree is much higher in Graph 2, for a particular k value more nodes are available for sharing content than in Graph 1. As we already pointed our earlier, in hop unrestricted case, the traffic savings is same for both Graph 1 and Graph 2. It is to be noted that traffic savings is independent of predictor accuracy and cache eviction policy.

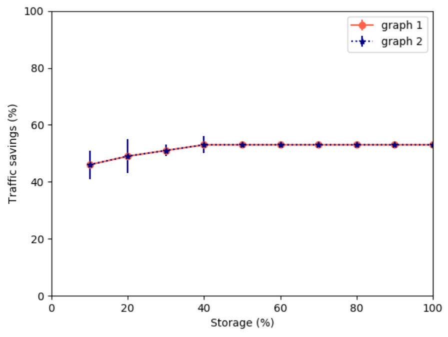
In Fig. 4.12b we study the impact of storage in traffic savings in a hop unrestricted network. We see that with 30% storage maximum possible traffic savings can be obtained for both Graph 1 and Graph 2, which is much lower than the storage needed in Scenario 1 (Section 4.2).

Average hopcount per sharing

We evaluate the average number of hops per sharing to study how close our proposed strategy has been able to put content to the interested user(s). As we can see from Fig. 4.13a, for $k \leq 3$, the average hopcount per sharing value is much lesser in Graph 1 than in Graph 2. This is because in case of a hop restricted network, in Graph 1 lesser content

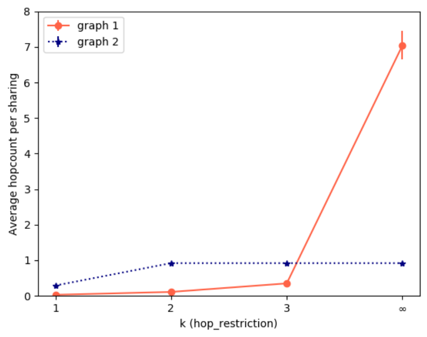


(a) Different hop restricted networks

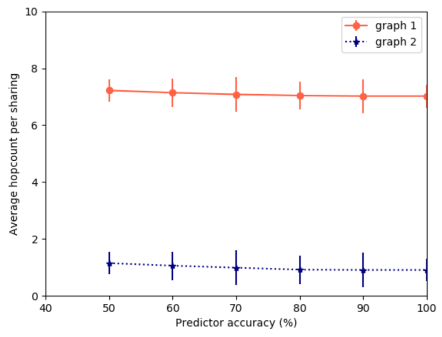


(b) Varying storage capacity in hop unrestricted network

Figure 4.12: Traffic savings in apartment scenario



(a) Different hop restricted networks



(b) Varying predictor accuracy in hop unrestricted network

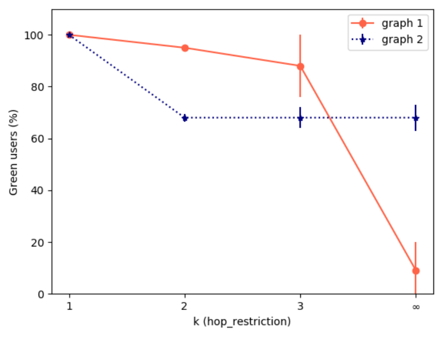
Figure 4.13: Average hopcount per sharing in apartment scenario

sharing takes place than in Graph 2. However, when the network becomes hop unrestricted, this value is much higher in case of Graph 1 than in case of Graph 2 because in Graph 2 average node degree value is higher so the average number of hops between any pair of nodes is lower.

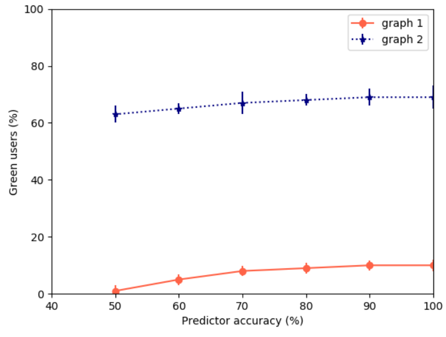
In Fig. 4.13b we study the impact of predictor accuracy in a hop unrestricted network. In both Graph 1 and Graph 2, the average hopcount value decreases as the predictor accuracy increases.

Green users

We evaluate the percentage of green users who get the content within 1 hop at maximum. We can see from Fig. 4.14a that in Graph 1, the percentage of green users decreases when the value of k increases and in Graph 2 this percentage value becomes constant for $k \geq 2$.

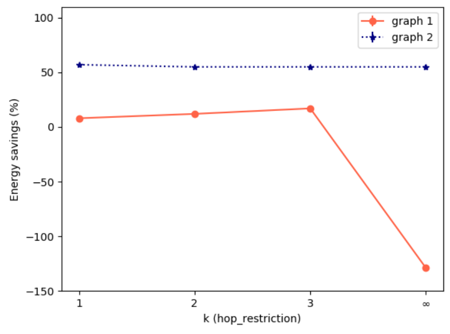


(a) Different hop restricted networks

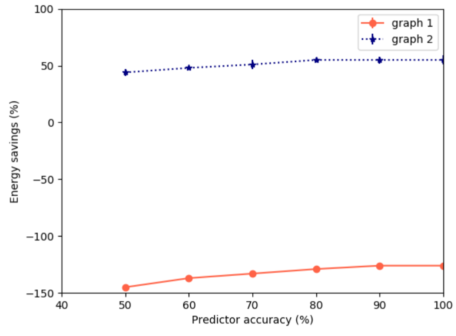


(b) Varying predictor accuracy in hop unrestricted network

Figure 4.14: Green users in apartment scenario



(a) Different hop restricted networks



(b) Varying predictor accuracy in hop unrestricted network

Figure 4.15: Energy savings in apartment scenario

From Fig. 4.14b we see that in an unrestricted network, when the value of predictor accuracy increases, for both Graph 1 and Graph 2 the percentage of green users also increases. This is because when the predictor accuracy increases, content are placed more closer to the interested users and more users get the content within a single hop.

Energy savings

We measure energy savings with respect to all the parameters, i.e., storage, predictor accuracy and cache eviction policy and study the impact of each of these parameters on energy savings. As we can see from Fig. 4.15a, in Graph 1 energy savings drops sharply when the network becomes hop unrestricted because the average hopcount per sharing value increases. In Graph 2 the energy savings becomes constant for $k \geq 2$ because the average hopcount per sharing value also becomes constant in Graph 2 for $k \geq 2$.

In case of a hop unrestricted network, energy savings increases when the predictor accuracy increases both in Graph 1 and Graph 2. This is quite expected because when the predictor accuracy increases, average hopcount per sharing value decreases which in turn increases the energy savings.

Observations

We make a few observations on the results generated using Graph 1 in this scenario in case of a hop unrestricted network:

- Average hopcount per sharing value is more than 7. In reality server will be reachable within lesser number of hops than 7 in most cases and so sharing will reduce.
- Percentage of green users falls below 20%, which is a very poor result.
- Energy savings is negative, which means getting content from the server consumes less energy than getting content from the peer. This is against the motivation behind our work.

Observing these results, we focus on *multiple content placements instead of a single one.* In the next Section we discuss several methods for multiple content placements in a network and compare their performances.

4.4 Multiple content placement

So far we considered placing content only once to the node with the highest Shapley value. But when the number of users are very high, say 1000, content sharing is not possible between any two nodes and thus it will not be enough just to place one content. We have to place multiple contents in the network so that traffic savings is high while the average hopcount per sharing value remains low. We follow four different methods for multiple content placement in the network.

4.4.1 Method descriptions

We describe four methods of multiple content placements:

Method 1

In this method, we at first decide in how many nodes we want to place the content. The number of nodes in which we want to place the content is expressed as a percentage of the total number of nodes in the network. For example, if there are n nodes in the network and we want to place content at *th* percent of nodes, we want to place content at $n_p = \frac{n \times th}{100}$ nodes. At first we calculate the Shapley values for all the nodes and sort them in decreasing order of their values. From the list, we select the first n_p values, find the node id corresponding to each of those values and place the content in that node.

Method 2

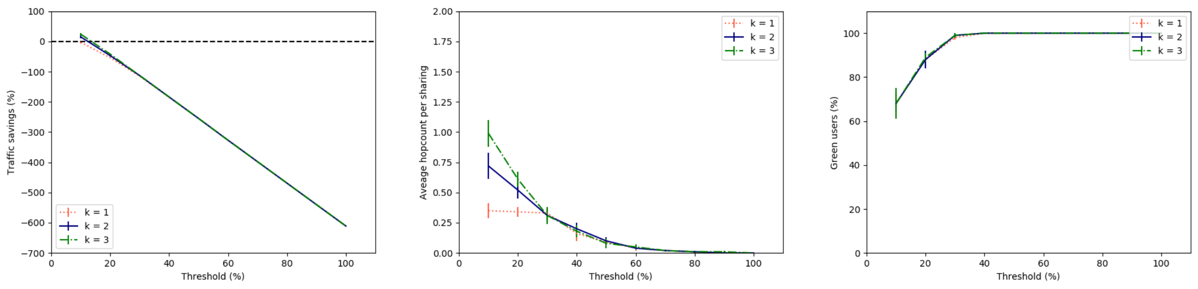
In this method, at first we calculate the Shapley values of all the nodes and sort them in decreasing order. Based on the the hop restricted network we consider, we set the value of k in our method. We also set the value of a threshold percentage like we did in Method 1. If there are n nodes in the network and we want to place content at *th* percent of nodes, we want to place content at $n_p = \frac{n \times th}{100}$ nodes. We always place at least 1 content in the node with the highest Shapley value. While considering the next node from the list, if this node is within k hops away from any of the nodes where the content has been placed, we skip that node. This way we keep on placing content in the nodes until content has been placed in n_p number of nodes.

Method 3

In this method we place content based on a threshold value. For a particular content we calculate Shapley value for all nodes in the network and we select the maximum Shapley value in the list. Let us denote it as *max_shapley _value.* We also define a threshold value *th* in percentage. We place content in all the nodes whose Shapley value is greater than or equal to $\frac{th \times max\_shapley\ \_value}{100}$.

Method 4

In this method at first we partition the graph using the greedy graph partitioning algorithm described in Section 3.2.5. based on the hop restricted network we consider, we correspondingly set the value of k (maximum number of hops) at the input of the algorithm. In each cluster at most one content is placed and it is shared by the rest of the members of the cluster. Here also we consider the same two parameters described in Method 2 - *max_shapley _value* and *th.* For each cluster, we consider the node with the highest Shapley value. If this Shapley value is greater than or equal to $\frac{th \times max\_shapley\ \_value}{100}$, we place the content in the cluster, otherwise we skip the cluster.



(a) Traffic savings (b) Average hopcount per sharing (c) Green users

Figure 4.16: Performance of Method 1 in network with 10 nodes

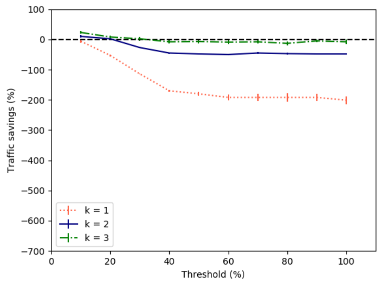
4.4.2 Performance evaluation

In this Section we evaluate the performance of these four methods described above in two different scenarios.

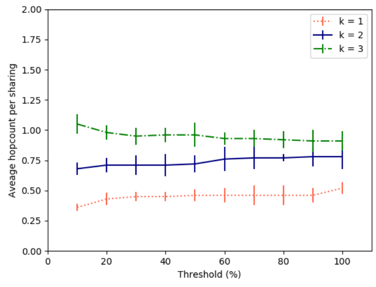
Sparsely connected network with 10 nodes

In this section we evaluate the performance of the four methods mentioned above. We create a random graph consisting of 10 nodes using Erdos-Renyi model and from the BBC iPlayer dataset we randomly select 10 ip addresses, and map each of them to one of the nodes randomly. We do these random mapping 10 times and evaluate the performance. We vary the number of hops (k) between 1 - 3.

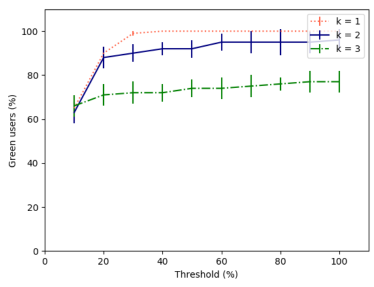
Traffic savings If we look at the plots Fig. 4.16a, 4.17a, 4.18a we see that for all three methods in some cases we get traffic savings in negative. That is because when we are placing content multiple times, it is not always happening that a node where the content is placed is going to watch it. For example, for some content which is watched only once by only one user, if that content is placed multiple times at different nodes, traffic savings becomes negative. For Method 1, from Fig. 4.16a we see that for all the k values, we get positive traffic savings only when the threshold value is 10%. Beyond that, we always get traffic savings in negative. The savings decreases when the threshold value increases because when the threshold value increases, more number of nodes are selected for content placement which in turn reduces the traffic savings. For Method 2, as we can see from Fig. 4.17a that traffic savings increases with increase in the threshold value. This is because when the threshold value increases, the minimum value which a node should have to be placed with a content increases, which means in lesser number of nodes content are being placed. This in turn increases the traffic savings. For this method, having a threshold value greater than or equal to 90% gives us positive traffic savings. We get the best traffic savings using Method 3. Here also the traffic savings increases with the threshold value



(a) Traffic savings

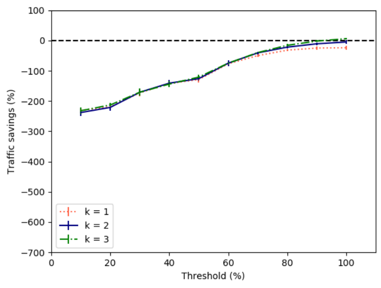


(b) Average hopcount per sharing

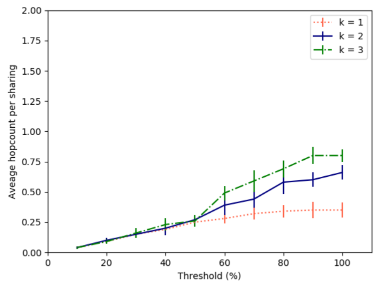


(c) Green users

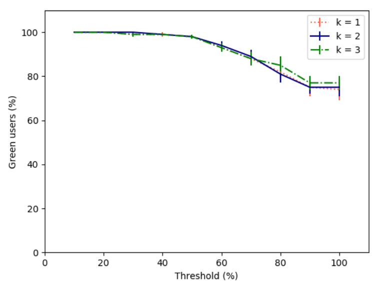
Figure 4.17: Performance for Method 2 in network with 10 nodes



(a) Traffic savings



(b) Average hopcount per sharing



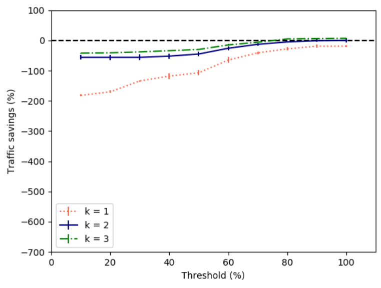
(c) Green users

Figure 4.18: Performance for Method 3 in network with 10 nodes

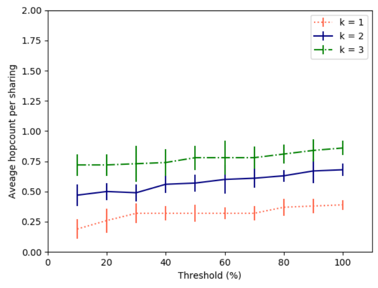
following the same reason as Method 2. We see that we get more traffic savings for a same threshold value when the k value is higher. This is for obvious reason, the more is the k value, the more is the sharing in the network. From Fig. 4.18a we see that using Method 3, we can get up to 25% of traffic savings (when the value of k is 3).

Average hopcount per sharing For all three methods, we see that for all k values the average hopcount per sharing value is less than or equal to 1. This proves that our strategy is really able to put the content closer to the interested users only. For Method 1, the hopcount value decreases with the increase in the threshold value as we can see in Fig. 4.16b. This is because the more is the threshold value, the more number of contents are placed in the network, and the average hopcount value becomes lesser. For Method 2 and Method 3, lesser number of contents are placed with the increase in threshold value, and that is why average hopcount value also increases (see Fig. 4.17b and 4.18b).

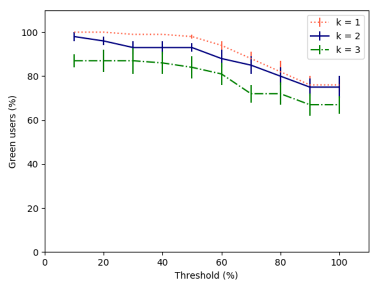
Green users For Method 1, more content are placed with the increase in threshold value, which in turn increases the percentage of green users in the network (Fig. 4.16c). For Method 2 and 3, lesser number of contents are placed with the increase in threshold



(a) Traffic savings

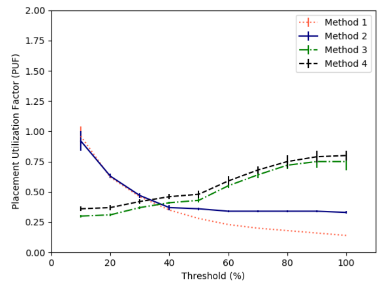


(b) Average hopcount per sharing

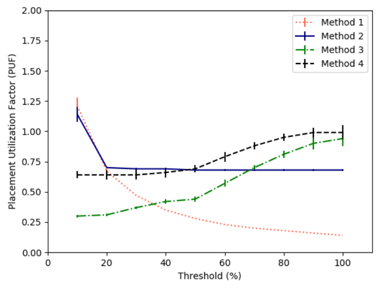


(c) Green users

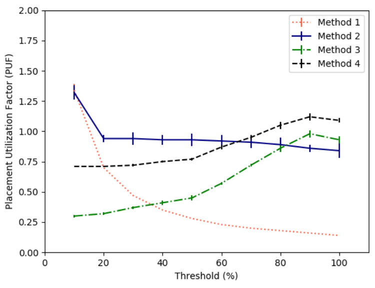
Figure 4.19: Performance for Method 4 in network with 10 nodes



(a) k=1



(b) k=2



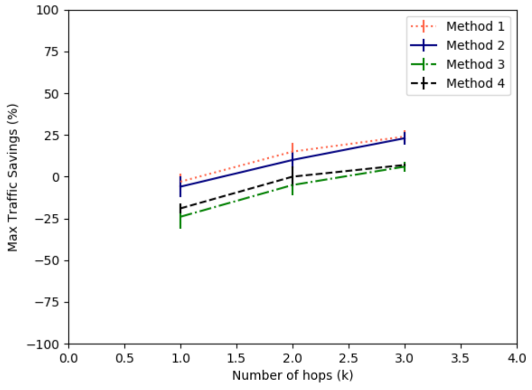
(c) k=3

Figure 4.20: Comparison of PUF values in network with 10 nodes

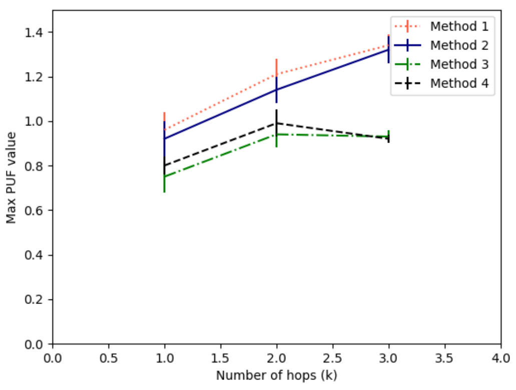
value, which in turn reduces the percentage of the green users (Fig. 4.17c and 4.18c).

Placement Utilization Factor (PUF) This is a new metric we introduce to display how effective how content placement method is. PUF is defined in the following way - if a content has been placed in the network p times and the placed contents have been shared s times by different users, then we define the Placement Utilization Factor (PUF) as $\frac{s}{p}$. It measures the average number of sharing per placed content. The more is the value of PUF, the more is the traffic savings and the better is the method. From Fig. 4.20 we see that for all k values, for Method 1 and Method 2 PUF value decreases with threshold value whereas for Method 3 and Method 4 PUF value increases with threshold value. This behavior is quite similar with the traffic savings behavior and happens for the same reason.

Selection of best method In the previous section we described four methods for multiple content placements. Via simulation we determined the best threshold values when we get the best results from these methods in terms of traffic savings and placement utilization. In Fig. 4.21 we plotted the best results we got from the previous section. For a particular hop restricted network (or, for a particular k value), we select the maximum

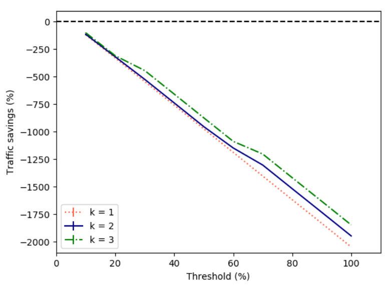


(a) Maximum traffic savings for different hop restricted network

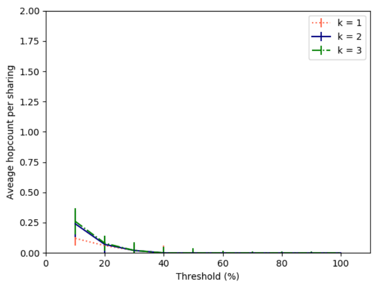


(b) Maximum PUF values for different hop restricted network

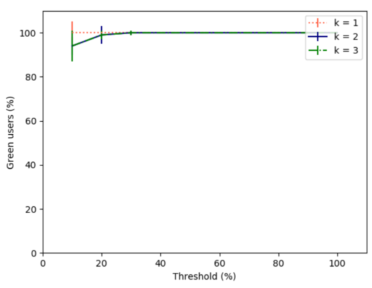
Figure 4.21: Maximum traffic savings and PUF values in network with 10 nodes



(a) Traffic savings



(b) Average hopcount per sharing



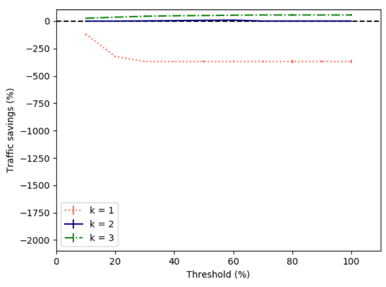
(c) Green users

Figure 4.22: Performance for Method 1 in network with 100 nodes

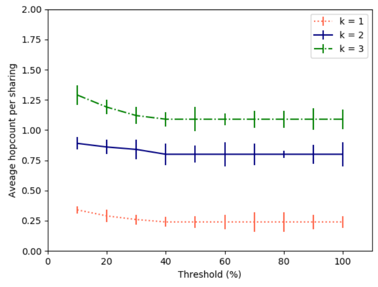
traffic savings we get from each of the methods and plot them in Fig. 4.21a. For all k values, we see that Method 1 gives the maximum traffic savings. Similarly for a particular k value we select the maximum PUF value we get from each of the methods and plot them in Fig. 4.21b. Here also we see that Method 1 always yields the maximum PUF values.

Densely connected network 100 nodes

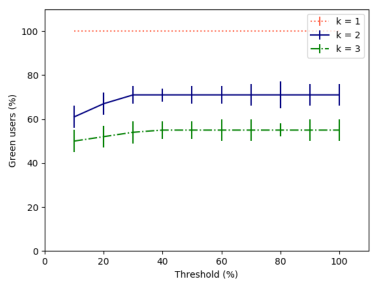
In this section we evaluate the performance of the four methods. We created a random graph consisting of 100 nodes using Erdos-Renyi model and from the BBC iPlayer dataset we randomly select 100 ip addresses, and map each of them to one of the nodes randomly. We do these random mapping 10 times and evaluate the performance. We vary the number of hops (k) between 1 - 3.



(a) Traffic savings

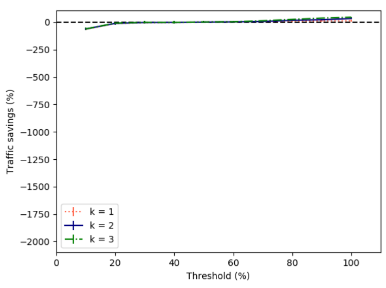


(b) Average hopcount per sharing

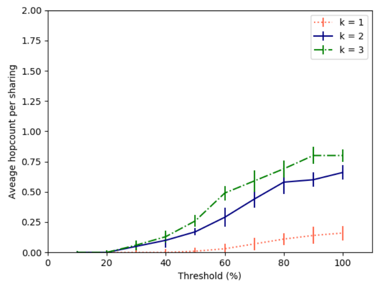


(c) Green users

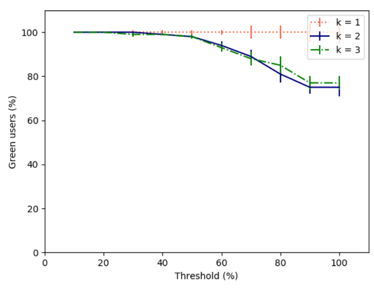
Figure 4.23: Performance for Method 2 in network with 100 nodes



(a) Traffic savings



(b) Average hopcount per sharing



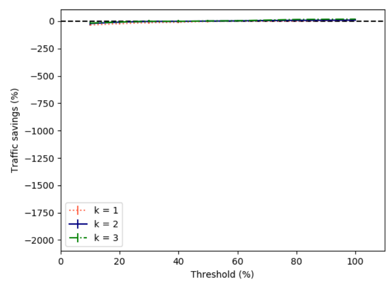
(c) Green users

Figure 4.24: Performance for Method 3 in network with 100 nodes

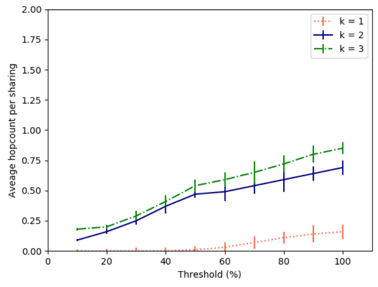
Traffic savings We can see from Fig. 4.22a that using Method 1 always yields a negative traffic savings in this scenario. Using Method 2 we get positive traffic savings only when k = 3 (Fig. 4.23a). Using Method 3 (Fig. 4.24a) and Method 4 (Fig. 4.25a) we get the best traffic savings in this scenario.

Average hopcount per sharing We see that average hopcount per sharing value is lowest in case of Method 1. This is quite understandable because using Method 1 we put the maximum number of copies of a content in the network, so the content is more available to any user than any other method which in turn reduces the average hopcount per sharing value.

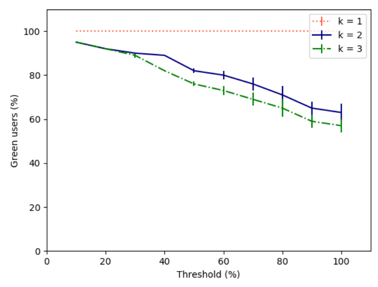
Green users We also see that we get the maximum number of green users when we use Method 1. This is also for the same reason that using Method 1 we put the maximum number of copies of a content in the network, so the content is more available to any user in lesser number of hops which in turn increases the percentage of green users in the network.



(a) Traffic savings

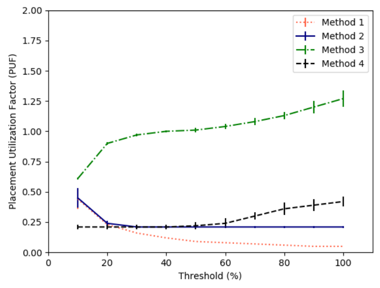


(b) Average hopcount per sharing

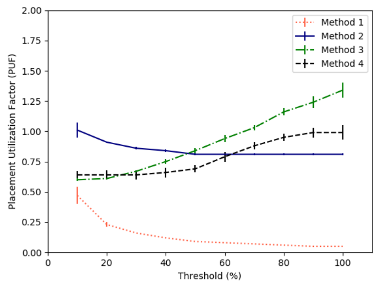


(c) Green users

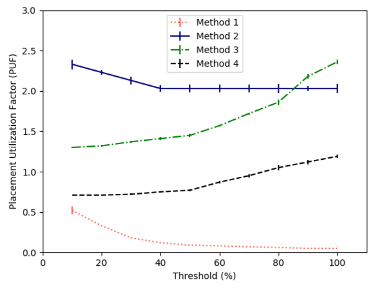
Figure 4.25: Performance for Method 4 in network with 100 nodes



(a) k=1



(b) k=2

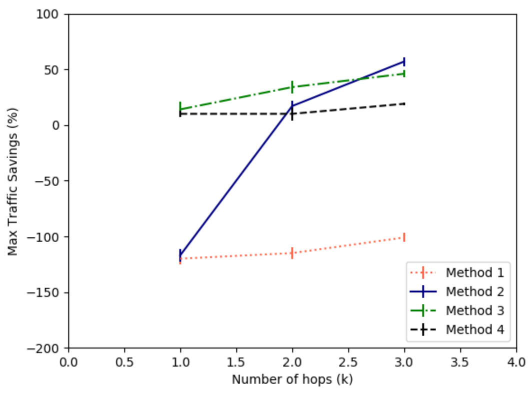


(c) k=3

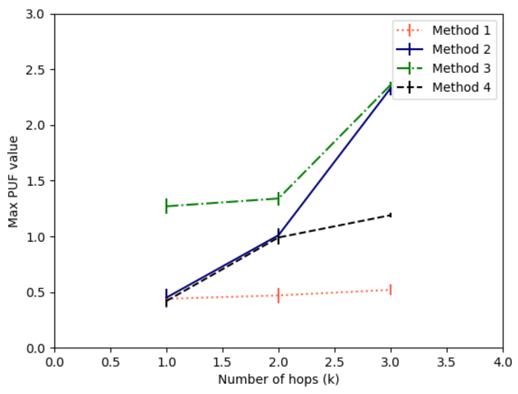
Figure 4.26: Comparison of PUF values in network with 100 nodes

Placement Utilization Factor (PUF) From Fig. 4.26 we see that for k =1 and 2, we get the best PUF values using Method 3. For k = 3, we get the best PUF values using both Method 2 and Method 3.

Selection of best method In Fig. 4.27 we plotted the best results we got from the previous section. For a particular hop restricted network (or, for a particular k value), we select the maximum traffic savings we get from each of the methods and plot them in Fig. 4.27a. For k = 1 and 2, we see that Method 3 gives the maximum traffic savings. Although for k = 3 Method 2 gives the maximum traffic savings, it yields very poor performance for k = 1 and 2. Method 3 gives the most consistent results in this respect. Similarly for a particular k value we select the maximum PUF value we get from each of the methods and plot them in Fig. 4.21b. Here also we see that Method 3 always yields the maximum PUF values. As we will be studying multiple content placements in dense urban scenario (densely connected graph with large number of nodes), we select Method 3 for content placement in that kind of scenario.



(a) Maximum traffic savings for different hop restricted network



(b) Maximum PUF values for different hop restricted network

Figure 4.27: Maximum traffic savings and PUF values in network with 100 nodes

4.5 Multiple content placement in Apartment scenario

In this Section we discuss about multiple content placement in an apartment scenario as described in Section 4.3.

4.5.1 Network formation

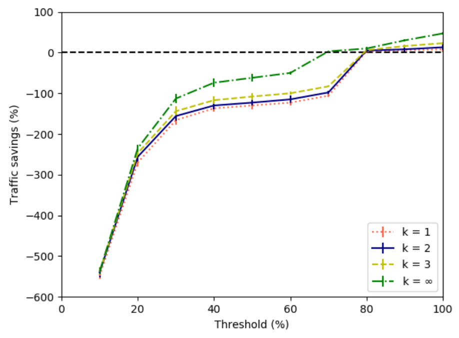
This is same as described in Section 4.3.1. We use the same set of values as described in Section 4.3.4 and generate two different graphs - Graph 1 and Graph 2 for evaluation purpose.

4.5.2 Real data mapping

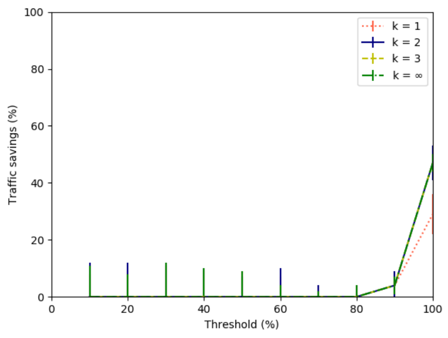
This is same as described in Section 4.3.2.

4.5.3 Content placement

The difference of this content placement with the one described in Section 4.3.3 is that here we discuss about multiple content placements. Content placement is done following Method 3 discussed in previous section.



(a) Traffic savings in Graph 1 in apartment scenario for multiple content placements



(b) Traffic savings in Graph 2 in apartment scenario for multiple content placements

Figure 4.28: Traffic savings in apartment scenario for multiple content placements

4.5.4 Measurement of performance

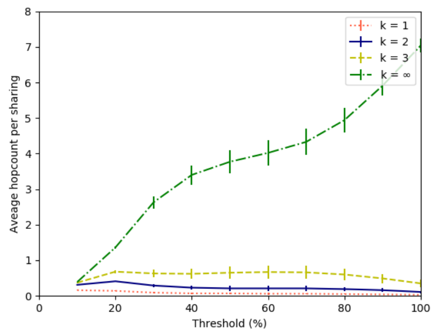
Performance measurement is done as the same way described in Section 4.3.4.

Traffic savings

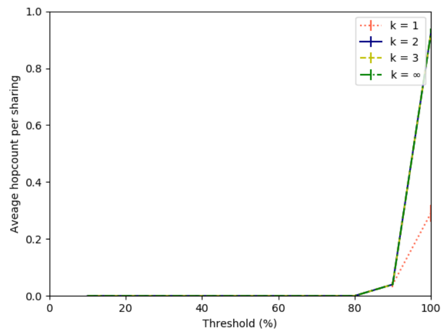
From Fig. 4.28a we see that in case of Graph 1, for low threshold values we get a negative value in the traffic savings. This is because more number of contents are placed than needed. When the threshold value becomes high, lesser number of copies of the same content are placed, and it in turn increases the traffic savings. In a hop unrestricted network, we always get more traffic savings as expected. However, in case of Graph 2, from Fig. 4.28b we see that for low threshold values we get a positive traffic savings with the value close to zero. In this case, traffic savings remains almost constant for low threshold values (¡ 80%) and has a sharp increase when the threshold varies between 80% - 100%.

Average hopcount per sharing

In case of Graph 1, average hopcount value is always greatest in case of a hop unrestricted network. In hop restricted network, this value almost remains constant when the threshold increases. For a certain threshold value, the average hopcount value is lower in case of Graph 2 than in case of Graph 1. This is because in Graph 2, the average node degree is higher so the graph is more well connected than Graph 1 which in turn reduces the average number of hops when a content is shared.

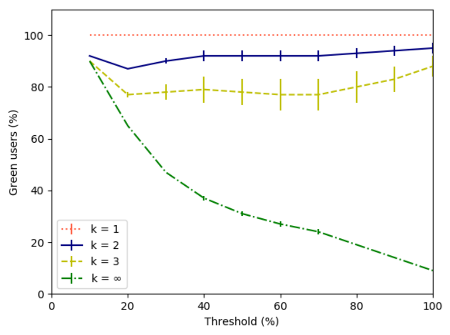


(a) Average hopcount per sharing in Graph 1 in apartment scenario for multiple content placements

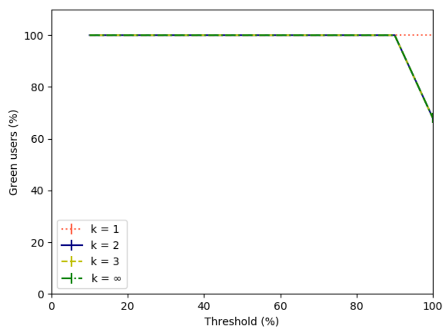


(b) Average hopcount per sharing in Graph 2 in apartment scenario for multiple content placements

Figure 4.29: Average hopcount per sharing in apartment scenario for multiple content placements

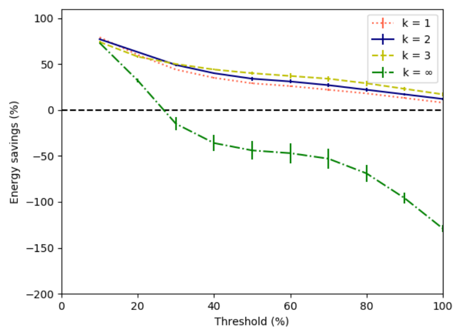


(a) Green users in Graph 1 in apartment scenario for multiple content placements

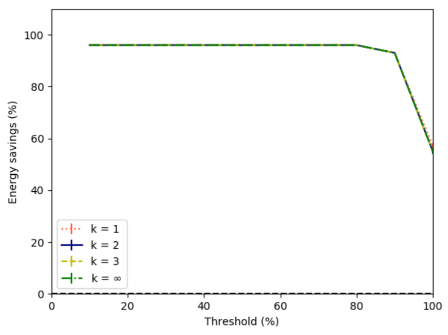


(b) Green users in Graph 2 in apartment scenario for multiple content placements

Figure 4.30: Green users per sharing in apartment scenario for multiple content placements



(a) Energy savings in Graph 1 in apartment scenario for multiple content placements



(b) Energy savings in Graph 2 in apartment scenario for multiple content placements

Figure 4.31: Energy savings in apartment scenario for multiple content placements

Green users

In case of Graph 1, for a hop unrestricted network, the percentage of green users decreases when the threshold value increases. For hop restricted networks, this value remains almost constant. In case of Graph 2, for low threshold values (¡ 80%), the percentage of green users remains almost constant and decreases when the threshold value varies between 80% - 100%.

Energy savings

In case of Graph 1, for a hop unrestricted network, when the threshold value is quite high, we get a negative energy savings. This is because the average hopcount value is so high that the fetching the content directly from the server includes lesser number of hops. In case of Graph 2 the behavior similar for all hop restricted networks.

Chapter 5

Content pricing policy

In this chapter we provide a model for content pricing in a content distribution model we described in the thesis. The work of this chapter has been submitted to the CTTE-FITCE conference, 2019 [13].

5.1 Background

There have already been some previous research works on the pricing and distribution of digital content in centralized and decentralized distribution model. In [43] the authors explore about Music as a Service and find out that next to price, contract duration and product qualities are two most important product attributes. Their research also studies configurations on consumers' utility and their willingness to pay (WTP) for premium offers. The paper [44] compares between two business models. In the first, free of access is provided with low product quality and advertisements, while in the second the access is charged and product is of high quality. It has been shown that if the users are advertisement tolerant, the first policy is more profitable. on the other hand, researchers in [45] studied the same thing and showed that companies are better off providing both pricing and advertising options to consumers. They suggested that as advertisement revenue rate increases, advertising level should be kept low and based on the factors that affect the price and advertising decisions, they derived the optimal price and advertising level. In [46] the researchers studied these two kind of policies (fee based and free with advertisement) in detail and showed that showed that the mix group that uses both free and fee strategies are the most profitables. It also highlighted that lowering the product's price to 0 will lead to price competitiveness among competitors, thus securing more consumers in the initial market. When more consumers are secured, it will be more advantageous regarding word-of-mouth effect and product diffusion. In [47] the researchers proposed a game theory based pricing policy for job allocation in mobile grid networks. They introduced a game theoretic framework to obtain pricing model between a Wireless Access Point (WAP) server and a mobile device.

Along with that, they also introduced a game theory based workload allocation scheme for the mobile devices to maximize the revenue. In [48] the authors analyzed the pricing schemes and DRM (digital right management) protection policy with respect to different collaborative market structures. As the piracy is closely associated with the objects and the distribution channel, they also examine the impact of content quality and network effects on the development of strategies based on all these factors. But this is a model for host centric network only, no discussion for peer-to-peer networks. In [49] the authors proposed a lottery based p2p pricing scheme. This model provides higher revenue for peers who intend to participate and contribute with other peers and higher cost for those who choose to behave selfishly. But the service provider is not included in the game, so the pricing policy of the content provider is not taken into account. The paper [50] compares between two models of digital content distribution - client-server model and peer-to-peer model. They described a monopolistic pricing scheme for distributing digital content over peer-to-peer networks that rewards peer users who participate in the distribution process and showed that peer-to-peer model is more profitable if pricing mechanism provides strong incentives to users to share content.

All the previous research works described above are based on the pricing and distribution of digital content in centralized as well as decentralized network model. In this paper we assume a hybrid content distribution model where a user can choose either the centralized or the decentralized model to get the content. For this model, we provide the model to determine the content pricing policy in different possible cases based on PP's interests. To the best of our knowledge, we are the first ones to come up with game theoretic pricing policy in such kind of hybrid model.

5.2 Basic Distribution Model

Fig. 5.1 depicts the steps of the process of content distribution: from the content provider (CP) at the top to the final user at the bottom. CPs are the owner of content (e.g., MGM, Walt Disney and other big production houses who make the videos and music). Platform Providers (PP) are the owner of the platform which helps in sharing the digital media content among the users (e.g., YouTube, Netflix). Network providers (NP) (e.g., Vodafone, Telefonica) provide the connectivity between providers and users. In the content distribution scenario, NPs distribute the content to the users through their network. Last but not least, content customer (CC) is the final user, who downloads the media content (e.g., watching a movie, listening to music). From this point onward, throughout the paper we will use the terms 'user' and 'CC' interchangeably.

Nowadays, users have contracts with local network providers, in order to guarantee their connectivity to get internet access. Furthermore, users also have contracts to one or more PP in order to get access to the content portfolio they offer (either their own content or from other CP as e.g., MGM), but in this paper we only focus on the interaction between

Notation	Description
f [Bytes]	File size
x	Number of users who download the file
y	Number of users who download the file via premium option
p_n [€/Byte]	Price paid by PP to network provider to deliver a content from server to client
p_b [€/Byte]	Price a user pays when the user downloads the content selecting premium option
p'_b [€/Byte]	Price a user pays when the user downloads the content selecting standard option
p_u [€/Byte]	Reward given to user by PP who shares the content in standard option
ϵ [€/Byte]	Energy cost per Byte when sharing the content with other users in standard option

Table 5.1: List of notations

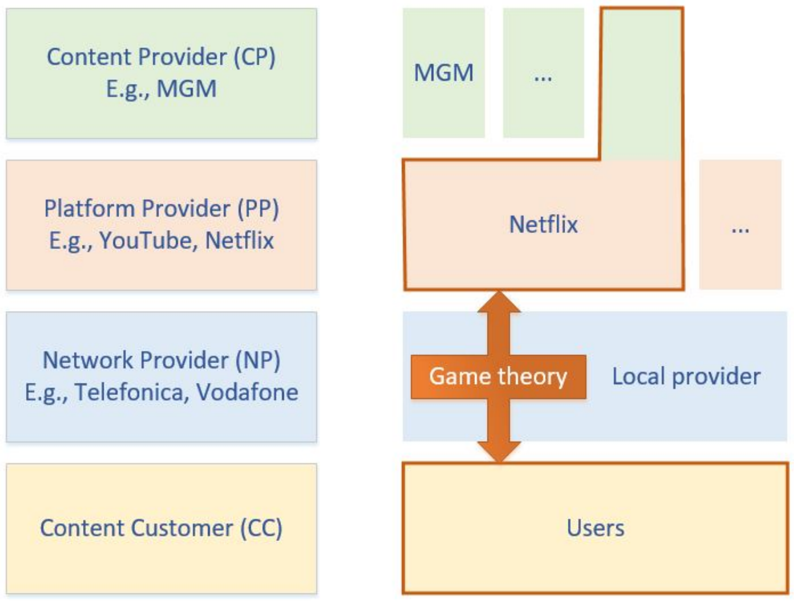


Figure 5.1: Top-down Content distribution: from content provider to user, with highlighted players of the game.

the PP and the user.

Let us denote x to be the total number of users who download the content c of a size of f Bytes. Users have two options:

- *Premium* service: This option guarantees the user to get the content from the PP server at a cost of p_b [€/Byte]. This user is shown in blue in Fig. 5.2.
- *Standard* service: This option allows the user to get the content from other *standard* users with lower delay in case it is available. Otherwise the user gets the content from the server. The cost to get the content is $p'_b < p_b$ [€/Byte]. Sharing the content with others also imply an extra energy consumption, which is proportional with the content size. Let us denote with ϵ [€/Byte], the energy cost per transmitted Byte. This user is shown in green in Fig. 5.2, highlighted when getting the content from the server and sharing it with the neighbors.

Let us consider, that from the set of x users, y select *Premium* option (hence, $x - y$ are the users selecting the the *Standard* option). Furthermore, in order to encourage the content sharing with other interested users, the PP also offers a compensation price p_u [€/Byte] every time that a shares the content with another user.

Last but not the least, the PP should also pay the Network Provider (NP) for the network resources used to distribute the content. Let us denote by p_n [€/Byte] the amount that NP charges the PP for distributing content c to a particular user u. It is safe to assume that the value of p_n is greater than 0 and remains constant for a particular c. This calculation

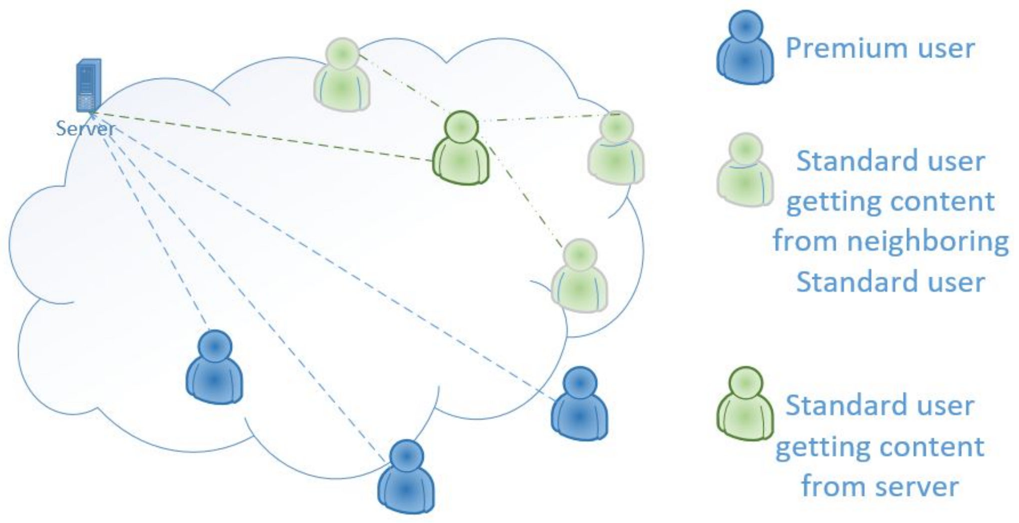

Server
Premium user
Standard user getting content from neighboring Standard user
Standard user getting content from server

Figure 5.2: Case example with $x = 7$ users: $y = 3$ and $x - y = 4$ users getting *Premium* and *Standard* access respectively.

is done on a per content basis - for different contents the values vary.

Let us assume that for a particular content c, the values of f, x and y are constant. The main goal of the PP and the user is to maximize their benefits, which are defined as the difference between the total generated revenues and the total cost. Hence, the problem aims at finding the best values of p_b, p'_b and p_u to maximize the benefits. In the next section we describe the game between the PP and user and show that equilibrium can be reached in such cases.

5.3 Game Theoretic Equilibrium

Let us consider three scenarios based on the value of y:

- $y = x$, i.e., all of the x users select the *Premium* option, and hence, the load on the server is reduced by 0.
- $y = 0$, all of the x users select *Standard* option. In this case, one user will get the content from the server and the others (i.e., $x - 1$) will get the content for the first user. In this case, the load on the server will be potentially reduced by a factor of $\frac{x-1}{x}$.
- $0 < y < x$, (as shown in Fig. 5.2 for the particular case of x=7 and y=3). In this more generic case, the load on the server is reduced by a factor of $\frac{x-y-1}{x}$.

Let us analyze each scenario and discuss on how equilibrium can be reached if it is possible.

5.3.1 Case 1 : $y = x$

In this case, all the x users interested in getting content c select the *Premium* option.

The PP has to pay a total amount of $f.p_n.x$ to the NP to deliver the content to all x users. In return, the PP earns a total amount of $f.p_b.x$. So, in this case, the total amount PP benefits is $Rev_{pp1} = f.(p_b - p_n).x$. Now, to have a positive benefit, p_b should be greater than p_n. As p_n is constant, the higher is the value of p_b, the higher is the benefit for PP.

Each user pays a price of $f.p_b$ and hence, the benefit is $Ben_{user1} = -f.p_b$. The benefit decreases for higher p_b.

In this case, an equilibrium (a settlement of price to which both PP and user agree) can be obtained if $p_b = p_n = 0$, which is very unrealistic. As no realistic equilibrium (settlement of price) is not possible, PP selects the value of p_b and the upper limit on this value can be set based on the customer's willingness to pay for the content.

5.3.2 Case 2 : $y = 0$

In this case, all of the x users select the *Standard* option. This option promotes content sharing and hence, one user gets the content from the server, and the remaining $x - 1$ get the content directly from him/her.

PP

In this case, the PP has to deliver the content to at least one user from whom all the other users $x - 1$ can share. The revenue of the PP is $f.p'_b.x$.

The cost to deliver the content to one CC is $f.p_n$. Furthermore, the PP has to give a reward to the user who is sharing the content with other interested users and thus helping in the distribution of the content. The amount of the reward is $f.p_u$ per sharing. So, the total amount of reward PP has to give is is $f.p_u.(x - 1)$ as the content is shared $(x - 1)$ times. So, in this case the total benefit generated by PP is :

$$Ben_{pp2} = f.p'_b.x - f.p_n - f.p_u.(x - 1) \tag{5.1}$$

User

When a user downloads the content, it has to pay a price of $f.p'_b$ to the PP. The user also has to pay the energy consumed while sharing the content with $x - 1$ users, which can be expressed as $f.\epsilon.(x - 1)$.

On the other hand, the user gets the reward from the PP, which is $f.p_u.(x-1)$ for sharing content $(x-1)$ times.

So, the total benefit of the user can be expressed by

$$Ben_{user2} = f.p_u.(x-1) - f.p'_b - f.\epsilon.(x-1) \tag{5.2}$$

It is important to note that selecting the *Standard* option while getting a content, gives to the user a chance to have positive benefit. This should motivate the user to choose this option over *Premium* option, which always generates a negative benefit.

For the particular case of $x = 1$, i.e., only 1 user watches the content and cannot share it with any other user, then, from Eq. 5.1, the total benefit of the PP becomes $Rev_{pp2} = f - (p'_b - p_n)$. Hence, as the PP would always ensure to generates positive benefit and avoid loosing money, the following condition should hold

$$p'_b > p_n. \tag{5.3}$$

Nash Bargaining Solution (NBS)

Here we propose a bargaining between PP and users in a situation where:

- the value of $x \geq 2$,
- there is a conflict of interest on agreement, and
- there is a possibility of concluding a mutually beneficial agreement.

Considering p_n a constant for this particular game, we can see from Eq. 5.1 that the PP wants to maximize the following function $F_{pp2} = (p'_b.x - p_u.(x-1))$. We also see from Eq. 5.2 that the user wants to maximize the following function $F_{user2} = (p_u.(x-1) - p'_b)$. If the PP wants to increase the value of p'_b and lower the value of p_u, it decreases the benefit of the user. On the other hand, if the user wants to increase the value of p_u and lower the value of p'_b, it decreases the benefit of the PP. This is a classic bargaining problem and by using NBS, each of them can settle to a value which lies exactly in the middle of their interests. So, in this case, to attain equilibrium, both PP and user should settle to a value of

$$F_{eq} = \Big(\frac{F_{pp2} + F_{user2}}{2}\Big) = \frac{p'_b(x-1)}{2}$$

instead of F_{pp2} and F_{user2} respectively.

This equilibrium value depends on p'_b only and is independent of p_u. So, at equilibrium, the PP's and user's benefits are

$$Ben_{pp2} = f.\Big(\frac{p'_b(x-1)}{2} - p_n\Big) \tag{5.4}$$

$$Ben_{user2} = f.\Big(\frac{p'_b(x-1)}{2} - \epsilon(x-1)\Big) \tag{5.5}$$

Settlement of Price

We have already seen from Eq. 5.3 that $p'_b > p$. We also know that $p_n > 0$ and the value of p_n depends on the network provider. The upper limit on p'_b can be set by studying the market and based on the customer's willingness to pay for the content. The benefits for both PP and users are now at equilibrium, as both of them try to optimize $\frac{p'_b(x-1)}{2}$.

Observation In order to reduce the load on the server , the PP should motivate users to opt for the *Standard* option. For that purpose, one option is to guarantee $p_b > p'_b$. From Eq. 5.3, we know that $p'_b > p_n$. Hence, we can state that $p_b > p'_b > p_n$. To ease our calculation, let us express p_b and p'_b in terms of p_n:

$$p_b = m.p_n \qquad p'_b = n.p_n \tag{5.6}$$

where m and n are considered to be positive integers with $m > n$. Thus, the minimum possible values of m and n are 3 and 2 respectively. Considering these values in the PP's benefit given in Eq. 5.4, the PP's benefits can be expressed as

$$Ben_{pp2} = f.p_n.\Big(\frac{n(x-1)}{2} - 1\Big)$$

Let us compare these benefits with the benefit that the PP would get in Case 1 (all users get *Premium* option Ben_{pp1}). Subtracting $Ben_{pp2} - Ben_{pp1}$, we get

$$\begin{aligned} Ben_{pp2} - Ben_{pp1} &= f.p_n.(m-1).x - f.p_n\Big(\frac{n.(x-1)}{2} - 1\Big) \\ =& f.p_n\Big(\frac{(m-n).x + (m-2).x + n + 2}{2}\Big) \end{aligned} \tag{5.7}$$

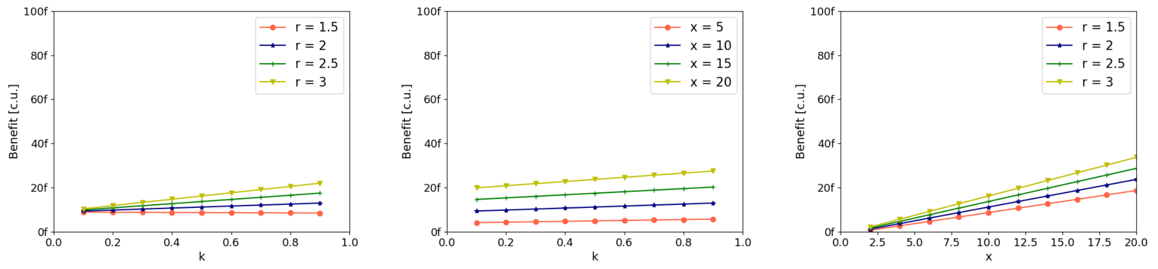
Since $m > n$ and $m > 2$, the value of Eq. 5.7 is always greater than 0, which means that the benefits generated in Case 1 are always higher than those in Case 2.

5.3.3 Case 3 : $x > y > 0$

This is the most general case, where from a group of x users getting content c, y users select the *Premium* option and the remaining $x - y$ select the *Standard* option. Here, as $y > 0$, so the lowest possible value of y is $y = 1$ and as $x > y$, the lowest possible value of x is $x = 2$.

PP

The costs of the PP in this case are $f.p_n.y + f.p_n + f.p_u(x - y - 1)$.



(a) Benefit vs k for different r (b) Benefit vs k for different x (c) Benefit vs x for different r

Figure 5.3: Ben_{pp3} dependence for varying parameters in a simple scenario.

In this case, the PP revenues are $f.p_b.y + f.p'_b(x - y)$.

The PP's benefits in this case are:

$$Ben_{pp3} = f.p_b.y + f.p'_b.(x - y) - f.p_u.(x - y - 1) - f.p_n.(y + 1)$$

User

The cost associated to the user selecting the *Standard* option and sharing the content with the remaining $x - y - 1$ users is: $f.p'_b + \epsilon.f.(x - y - 1)$. On the other hand, the revenue of such a user is: $f.p_u.(x - y - 1)$. The user's benefits in this case can be expressed by:

$$Ben_{user3} = f.p_u.(x - y - 1) - f.p'_b - \epsilon.f.(x - y - 1)$$

NBS

In this case, the PP wants to maximize $F_{pp3} = p_b.y + p'_b.(x - y) - p_u.(x - y - 1)$, whereas the user wants to maximize $F_{user3} = p_u.(x - y - 1) - p'_b$. As there is again a conflict of interest, using NBS both of them come to an agreement and equilibrium is reached. Proceeding the same way as described in Section 5.3.2, the equilibrium benefits for PP and users are:

$$Ben_{pp3} = f\left(\frac{p_b.y + p'_b.(x - y - 1)}{2} - p_n.(y + 1)\right) \tag{5.8}$$

$$Ben_{user3} = f.\left(\frac{p_b.y + p'_b.(x - y - 1)}{2} - \epsilon.(x - y - 1)\right) \tag{5.9}$$

Settlement of Price

The benefits of both PP and users are maximized when the term $f.\left(\frac{p_b.y+p'_b.(x-y-1)}{2}\right)$ is maximum. We use Lagrange multiplier [51] to obtain the maximum value of this expression with the constraint $p_b > p'_b$. Let us assume $p_b - p'_b = \delta$ where $\delta \in \mathbb{R}^+$. We introduce a new variable called Lagrange multiplier (λ) and study the Lagrange function defined by

$$\mathcal{L} = f.(p_b.y + p'_b.(x-y-1)) + \lambda.(p_b - p'_b - \delta)$$

The maximum is obtained by setting the derivatives (Eq. 5.10) to zero.

$$\frac{\partial \mathcal{L}}{\partial p_b} = f.y + \lambda \qquad \frac{\partial \mathcal{L}}{\partial p'_b} = f.(x-y-1) - \lambda \tag{5.10}$$

This leads to $\lambda_{max} = -fy$, and considering λ_{max}, to $f.(x-y-1) = -f.y$, i.e., $x = 1$, which contradicts the basic condition as we already stated that lowest possible value of x in this case is 2. This proves that we cannot have a maximum value for $p_b.y + p'_b.(x-y-1)$ with the constraints $x \geq 2$, $y \geq 1$, $x > y$ and $p_b > p'_b$.

Let us discuss how Ben_{pp3} varies in a simple example. Let us define r as the ratio between *Premium* and *Standard* service costs, i.e., $r = \frac{p_b}{p'_b} = \frac{m}{n}$. Let us define as k, the percentage of users getting the *Premium* service, i.e., $y = k.x$. In this case, Ben_{pp3} can be expressed as

$$Ben_{pp3} = \frac{f.p_n}{2}\Big(n.(r.k.x - x - k.x - 1) - (k.x + 1)\Big)$$

Let us evaluate the dependence of Ben_{pp3} with respect r, k and x. We use cost units (c.u.) as the basic unit of the p_n cost when transmitting $f = 1$ Byte. In all the simulations we assume $x = 10$ and $n = 2$ unless stated otherwise.

Fig. 5.3a shows the variation of Ben_{pp3} with respect k and r. It can be observed that for small k, i.e., small number of *Premium* users, r does not influence the benefits. However, as the percentage of *Premium* users increases, the impact of r also increases (the higher r, the higher the increase). Fig. 5.3c evaluates Ben_{pp3} with respect k and x for $r = 2$. The figure shows that x plays a major role since even for low k values, the benefits increases significantly with x and they do not vary substantively with k. Lastly, Fig. 5.3b depicts the benefits for different r and x values, keeping $k = 0.5$. It can be observed that small x limits the benefits independently of r. However, for large numbers of users, the benefits increases and more significantly, for high r values.

5.4 Pricing Policy

The PP has two main important objectives to consider

- maximize the benefits (Ben_{pp}), and
- reduce the load on the server (LR).

The PP's benefits and the server load reduction for the three previous cases presented in Section 5.3, have been summarized in Table 5.2.

	Ben_{pp}	LR
Case 1	$f.(p_b - p_n).x$	0
Case 2	$f.\frac{p'_b.(x-1)-2p_n}{2}$	$\frac{x-1}{x}$
Case 3	$f.\frac{p_b.y+p'_b.(x-y-1)-2p_n(y+1)}{2}$	$\frac{x-y-1}{x}$

Table 5.2: PP's benefits and load reduction for the three cases

The PP has to prioritize one of these two objectives, because as we have already seen in Case 1, the benefits for PP are maximized when the reduction of load on the server is minimized (i.e., $LR = 0\%$). The lesser the load on the server, the lesser is the maintenance cost. Let us then assume, for simplicity, a proportional model of the extra benefits with respect the load reduction: A load reduction of a factor of g, implies an increase of PP's benefit equal to ϕg, where $\phi > 0$ is the proportionality constant. At the end of this section we discuss how the value of ϕ can be calculated.

Let us also assume that PP wants to balance between these two factors, and accordingly the PP gives weights α and β to benefits and load respectively, such that $\alpha+\beta = 1$. In this way, the PP wants to maximize his utilization by maximizing the following target function $TF_{pp} = \alpha.Ben_{pp} + \beta.\phi.LR$. The value of the weights - α and β, solely depend on the PP. Let us consider three scenarios based on the value of α and β:

5.4.1 Scenario 1 : $\alpha = 0$, $\beta = 1$

In this scenario, the PP only aims at reducing the load on its server. According to Table 5.2, the load reduction is maximized in Case 2 and minimized for Case 1.

5.4.2 Scenario 2: $\alpha = 1$, $\beta = 0$

This scenario is considered when the PP wants to maximize the benefits. This case is the one discussed in detail in Section 5.3, along with the p_b and p'_b values for the three different cases.

5.4.3 Scenario 3: $\alpha > 0$, $\beta > 0$ and $\alpha + \beta = 1$

In Section 5.3 we discussed the values of p_b and p'_b in three different cases. However, when the PP launches a new media content, the PP does not know in advance the values of x and y. So, the PP should have such a policy such that the value of its TF_{pp} always remains the same and the value if independent of the values of x and y. It happens only if the value of the TF_{pp} is same in all of the three cases.

If we equalize the TF_{pp} of Case 1 and Case 2, we get:

$$\begin{aligned}\alpha.f.p_b.x &= \alpha.f.p'_b.\frac{(x-1)}{2} \\ &+ (\alpha.p_n.x.f - \alpha.p_n.f + \beta.\phi.\frac{(x-1)}{x})\end{aligned} \tag{5.11}$$

Similarly equalizing the TF_{pp} of Case 2 to Case 3 and combining it with Eq. 5.11, we obtain

$$p_b = p'_b + 2p_n + \frac{2\beta\phi}{\alpha f x} \tag{5.12}$$

As $p_n > 0, \beta > 0, \alpha > 0$ and $f > 0$, $p_b > p'_b$, which is the basic motivation of our solution to pursue more users to choose the standard option. If we consider that the large number of users get the content, i.e., large value of x, we can approximate $\frac{1}{x} \to 0$. Thus, $p_b = p'_b + 2.p_n$, and considering Eq. 5.6, it implies

$$m = n + 2 \tag{5.13}$$

Since $m = r.n$ with $n \geq 2$ and $r \geq 1.5$, $n.(r-1) = 2$. In this case, n is maximum when $(r-1)$ is minimum. As $(r-1)_{min} = 0.5$, we get $n_{max} = 4$ and $m_{max} = 6$. To summarize, pricing in *Premium* and *Standard* option should be, according to Eq. 5.6 with $2 \leq n \leq 4$ and $3 \leq m \leq 6$. So, all possible values of (m, n) are $\{(4,2),(5,3),(6,4)\}$.

On the other hand, putting this p_b value into Eq. 5.11 we get

$$\begin{aligned}&\alpha.f.(p'_b + 2p_n).x \\ =&\alpha.f.p'_b.\frac{(x-1)}{2} + \alpha.p_n.f(x-1) + \beta.\phi.\frac{(x-1)}{x}\end{aligned}$$

As the value of x is quite high, we can write $x - 1 \approx x$ and $x + 1 \approx x$. After rearranging the terms, we find

$$\begin{aligned}p'_b + 2p_n &= \frac{1}{2}p'_b + p_n + \frac{\beta\phi}{\alpha f} \\ \text{or, } p'_b &= -2p_n + 2\frac{\beta\phi}{\alpha f}\end{aligned} \tag{5.14}$$

Putting this value of p'_b in Eq. 5.12 we get $p_b = 2\frac{\beta\phi}{\alpha f}$. Hence, the values of p_b and p'_b are independent of x. Also,

$$\frac{1}{r} = \frac{p'_b}{p_b} = 1 - \frac{p_n \alpha f}{\beta\phi}$$

Based on all possible values of $\frac{p'_b}{p_b}$, once the values of f, p_n, α and β are known, the value of ϕ can also be easily calculated.

Chapter 6

Conclusions and Outlook

In this thesis, we provide a complete operational framework (content placement and pricing) from a content provider's perspective. We proposed a proactive caching strategy for a group of users who are able to share content among themselves using Wi-Fi. We used game theoretic centrality to find the most suitable node in the network for caching a content and discussed the advantages of using proposed approach over the topology based centrality metrics (degree, betweenness, eigenvector and closeness). To evaluate the performance of our proposed strategy, we analyzed the content access pattern of BBC iPlayer which consisted of 21 million users of United Kingdom with 231 million sessions and studied in a real life scenarios for users' homes. We study the performance of our proposed strategy in a grid based street model which resembles real life scenario. We consider two different types of buildings - singlehouse (in each building there is only one router) and apartments (in each building there are multiple apartments and in each apartment there is a router, so in each building there are multiple routers). We evaluate four metrics - traffic savings, energy savings, average hopcount per sharing and green users. In the singlehouse scenario, we found that a 49% savings in the traffic can be obtained using a storage capacity of 40% but when the routers were densely connected, almost same amount of savings can be obtained by using a storage capacity of 30%. Despite having the dependence on the accuracy of the predictor regarding calculation of the UIP values, we also found that our proposed strategy always places the content close to the potentially interested user(s) and on an average a user could find the content he was looking for within 0-4 hops.

Along with the content placement, we provide a content pricing policy also from the content provider's perspective. The content provider has mainly two objectives: maximize the benefits and reduce the server load. The load on the server can be reduced if users share content among themselves using the peer-to-peer network formed by them. In this thesis we provide a pricing policy from the content provider's perspective in three different cases and analyze the influence of different parameters to the prices. We used Nash Bargaining Solution (NBS) to determine the prices and showed that these values are in Nash equilibrium.

There are a few directions in which the work on content placement can be extended:

- *Payoff definition*: We define the payoff function so that content is placed close to the interested user(s) and sharing becomes maximum. To optimize different parameters like storage or link utilization, payoff function can be defined differently and the performance can be studied.
- *Cluster forming*: We considered only a particular scenario to evaluate our proposed strategy. It is better to have a clustering algorithm which can group users with similar interests in a densely populated urban scenario.

Similarly, the work on content pricing can be extended by considering the multiple content caching and sharing problem.

Bibliography

[1] M. Parameswaran et al. P2P networking: an information sharing alternative. *Computer*, 34(7):31–38, 2001.

[2] The blockchain-based digital content distribution system, author=Kishigami, J. and others. In *2015 IEEE Fifth Int. Conf. on Big Data and Cloud Computing*, pages 187–190. IEEE, 2015.

[3] C. Labovitz et al. Internet inter-domain traffic. *ACM SIGCOMM Computer Communication Review*, 41(4):75–86, 2011.

[4] Guoqiang Zhang, Yang Li, and Tao Lin. Caching in information centric networking: A survey. *Computer Networks*, 57(16):3128–3141, 2013.

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

[6] Aravindh Raman, Nishanth Sastry, Arjuna Sathiaseelan, Jigna Chandaria, and Andrew Secker. Wi-Stitch: Content delivery in converged edge networks. In *Proceedings of the Workshop on Mobile Edge Communications*, pages 13–18. ACM, 2017.

[7] Wei Koong Chai, Diliang He, Ioannis Psaras, and George Pavlou. Cache "less for more" in information-centric networks. In *International Conference on Research in Networking*, pages 27–40. Springer, 2012.

[8] Ioannis Psaras, Wei Koong Chai, and George Pavlou. Probabilistic in-network caching for information-centric networks. In *Proceedings of the second edition of the ICN workshop on Information-centric networking*, pages 55–60. ACM, 2012.

[9] Anubhab Banerjee, Nishanth Sastry, and Carmen Mas Machuca. Sharing content at the edge of the network using game theoretic centrality. In *2019 21th International Conference on Transparent Optical Networks (ICTON)*. IEEE, Jul 2019.

[10] P Series. Propagation data and prediction methods for the planning of short-range outdoor radiocommunication systems and radio local area networks in the frequency range 300 MHz to 100 GHz. Technical report, ITU, Tech. Rep. ITU-R, 2015.

[11] Recommendation ITU-R. P. 1238-7. *Propagation data and prediction methods for the planning of indoor radio communication systems and radio local area networks in the frequency range*, 900, 2012.

[12] J. F. Nash Jr. The bargaining problem. *Econometrica: Journal of the Econometric Society*, pages 155–162, 1950.

[13] A. Banerjee and C. Mas Machuca. Game theoretic pricing policy in online media delivery platforms. submitted in CTTE-FITCE 2019.

[14] Shuo Wang, Xing Zhang, Yan Zhang, Lin Wang, Juwo Yang, and Wenbo Wang. A survey on mobile edge networks: Convergence of computing, caching and communications. *IEEE Access*, 5:6757–6779, 2017.

[15] Mingyue Ji, 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(1):176–189, 2016.

[16] Jun Li, He Chen, Youjia Chen, Zihuai Lin, Branka Vucetic, and Lajos Hanzo. Pricing and resource allocation via game theory for a small-cell video caching system. *IEEE Journal on Selected Areas in Communications*, 34(8):2115–2129, 2016.

[17] Bo Bai, Li Wang, Zhu Han, Wei Chen, and Tommy Svensson. Caching based socially-aware D2D communications in wireless content delivery networks: a hypergraph framework. *IEEE Wireless Communications*, 23(4):74–81, 2016.

[18] Ejder Baştuğ, Mehdi Bennis, and Mérouane Debbah. Living on the edge: The role of proactive caching in 5G wireless networks. *arXiv preprint arXiv:1405.5974*, 2014.

[19] Engin Zeydan, Ejder Bastug, Mehdi Bennis, Manhal Abdel Kader, Ilyas Alper Karatepe, Ahmet Salih Er, and Mérouane Debbah. Big data caching for networking: Moving from cloud to edge. *IEEE Communications Magazine*, 54(9):36–42, 2016.

[20] Dmytro Karamshuk, Nishanth Sastry, Mustafa Al-Bassam, Andrew Secker, and Jigna Chandaria. Take-Away TV: Recharging Work Commutes With Predictive Preloading of Catch-Up TV Content. *IEEE Journal on Selected Areas in Communications*, 34(8):2091–2101, 2016.

[21] Dong Liu, Binqiang Chen, Chenyang Yang, and Andreas F Molisch. Caching at the wireless edge: design aspects, challenges, and future directions. *IEEE Communications Magazine*, 54(9):22–28, 2016.

[22] John Tadrous, Atilla Eryilmaz, and Hesham El Gamal. Proactive content download and user demand shaping for data networks. *IEEE/ACM Transactions on Networking (TON)*, 23(6):1917–1930, 2015.

[23] Ejder Baştuğ, Kenza Hamidouche, Walid Saad, and Merouane Debbah. Centrality-based caching for mobile wireless networks. In *1st KuVS Workshop on Anticipatory Networks*, 2014.

[24] Loomis C.P Proctor C.H. Analysis of sociometric data. In: Holland. *Research Methods in Social Relations*, pages 561–586, 1951.

[25] Linton C Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.

[26] Phillip Bonacich. Power and centrality: A family of measures. *American journal of sociology*, 92(5):1170–1182, 1987.

[27] Alex Bavelas. Communication patterns in task-oriented groups. *The Journal of the Acoustical Society of America*, 22(6):725–730, 1950.

[28] Anwar Said, Syed Shah, Hasan Farooq, Adnan Mian, Ali Imran, and Jon Crowcroft. Proactive Caching at the Edge Leveraging Influential User Detection in Cellular D2D Networks. *Future Internet*, 10(10):93, 2018.

[29] RHA Lindelauf, HJM Hamers, and BGM Husslage. Cooperative game theoretic centrality analysis of terrorist networks: The cases of jemaah islamiyah and al qaeda. *European Journal of Operational Research*, 229(1):230–238, 2013.

[30] Steven Tadelis. *Game theory: an introduction*. Princeton University Press, 2013.

[31] L. Shapley. A value for n-person games. *Annals of Mathematics Studies*, 28:307–317, 1953.

[32] Guillermo Owen. Multilinear extensions of games. *Management Science*, 18(5-part-2):64–79, 1972.

[33] Dennis Leech. Computing power indices for large voting games. *Management Science*, 49(6):831–837, 2003.

[34] Gilad Zlotkin and Jeffrey S Rosenschein. Coalition, cryptography, and stability: Mechanisms for coalition formation in task oriented domains. In *AAAI*, volume 432, page 437, 1994.

[35] Irwin Mann. *Values of large games, iv: Evaluating the electoral college by montecarlo techniques*. Rand Corporation, 1960.

[36] Shaheen S Fatima, Michael Wooldridge, and Nicholas R Jennings. A linear approximation method for the shapley value. *Artificial Intelligence*, 172(14):1673–1699, 2008.

[37] Paul Erdos and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

[38] Gianfranco Nencioni, Nishanth Sastry, Gareth Tyson, Vijay Badrinarayanan, Dmytro Karamshuk, Jigna Chandaria, Jon Crowcroft, Gianfranco Nencioni, Nishanth Sastry, Gareth Tyson, et al. SCORE: Exploiting global broadcasts to create offline personal channels for on-demand access. *IEEE/ACM Transactions on Networking (TON)*, 24(4):2429–2442, 2016.

[39] Eixample — Wikipedia, the free encyclopedia, 2019.

[40] Vytautas Valancius, Nikolaos Laoutaris, Laurent Massoulié, Christophe Diot, and Pablo Rodriguez. Greening the internet with nano data centers. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 37–48. ACM, 2009.

[41] Brian W Kernighan and Shen Lin. An efficient heuristic procedure for partitioning graphs. *Bell system technical journal*, 49(2):291–307, 1970.

[42] Charles M Fiduccia and Robert M Mattheyses. A linear-time heuristic for improving network partitions. In *19th Design Automation Conference*, pages 175–181. IEEE, 1982.

[43] Jonathan D. et al. Pricing of Content Services - An Empirical Investigation of Music as a Service. In *AMCIS/SIGeBIZ*, 2010.

[44] T. P. Thomes. An economic analysis of online streaming music services. *Information Economics and Policy*, 25(2):81–91, 2013.

[45] M. Fan et al. Selling or advertising: Strategies for providing digital media online. *Journal of Management Information Systems*, 24(3):143–166, 2007.

[46] H. S. Na et al. Efficiency comparison of digital content providers with different pricing strategies. *Telematics and Informatics*, 34(2):657–663, 2017.

[47] P. Ghosh et al. A game theory based pricing strategy for job allocation in mobile grids. In *18th Int. Parallel and Distributed Processing Symposium, Proc..*, page 82. IEEE, 2004.

[48] Y.-M. Li and Ch.-H. Lin. Pricing schemes for digital content with DRM mechanisms. *Decision Support Systems*, 47(4):528–539, 2009.

[49] M. Zghaibeh and F. C. Harmantzis. A lottery-based pricing scheme for peer-to-peer networks. *Telecommunication Systems*, 37(4):217–230, 2008.

[50] K. R. Lang and R. Vragov. A pricing mechanism for digital content distribution over computer networks. *Journal of Management Information Systems*, 22(2):121–139, 2005.

[51] JL Lagrange et al. Mécanique Analytique sect. IV 2 vol, 1811.