AN ABSTRACT OF THE THESIS OF

Chittibabu Tirupathi for the degree of Master of Science in Computer Science presented on November 5, 2019.

Title: Towards Long Short Term Memory Based Proactive In-Network Caching for Cloud RANs

Abstract approved: _____

Bechir Hamdaoui

The rapid population growth in large urban cities has led to an unprecedented increase in both the number and the diversity of wireless devices and applications with varying quality of service requirements in terms of latency and data rates. LinkNYC is an example of an urban communication network infrastructure, which replaces all the payphones in the five boroughs of New York City (NYC) with kiosk-like structures, called Links, with the goal of bringing fast and free public Wi-Fi access to thousands of city users. When enabled with data storage capability, these Links can play the role of edge cloud devices to allow in-network caching of popular Internet content to reduce access delay and backhaul traffic congestion by placing content closer to the end users. In this thesis, we propose k-means clustering to optimize content placement at the BSs, and we do so by clustering BSs based on the transmission delay between BSs and by proactively caching content at the clustered BSs with the aim of reducing content access delay and traffic congestion. Our proposed scheme also uses the Long Short Term Memory (LSTM)'s Seq-to-Seq model to predict content popularity at each BS. Using the LinkNYC network topology as the use case, we show through simulations that the proposed scheme reduces content access delay by minimizing content miss ratios and by reducing in-network content redundancy. Our study shows that our hybrid approach of proactive and reactive caching coupled with LSTM based popularity prediction provides potential solutions for fulfilling growing demands in urban communication networks.

Towards Long Short Term Memory Based Proactive In-Network Caching for Cloud RANs

by

Chittibabu Tirupathi

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented November 5, 2019
Commencement June 2020

Master of Science thesis of Chittibabu Tirupathi presented on November 5, 2019.

APPROVED:

Major Professor, representing Computer Science

Head of the School of Electrical Engineering and Computer Science

---

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Chittibabu Tirupathi, Author

ACKNOWLEDGEMENTS

I would first like to thank my thesis advisor Prof. Bechir Hamdaoui at Oregon State University. The door to Prof. Hamdaoui office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this thesis to be my own work, but steered me in the right direction whenever he thought I needed it. I would also like to thank my committee members, Prof. Ben lee, Prof. Rakesh Bobba and Prof. Margaret Louise Niess, who were involved in the validation process of my thesis. Without their passionate participation and input, the validation could not have been successfully conducted. Finally, I must express my very profound gratitude to my parents and to my brothers and sister for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

Chapter 1: Introduction

Studies [38] reveal that wireless and mobile devices carry about 71% of the total IP traffic & Internet video, and that TV shows will increase three-fold by 2022. In addition, social media (Facebook, Twitter, Instagram etc.) and multimedia services (YouTube, Netflix, etc.) have led to an exponential growth in capacity demand in mobile wireless networks. To cope with these growing service and bandwidth demands, mobile communication networks have resorted to deploying various wireless communication & networking technologies, including cognitive radio and dynamic spectrum sharing [5–9, 20–27, 29, 31, 32, 36, 46, 47, 50–55, 63, 69, 70], MIMO [33–35], and wideband/mmWave spectrum access [30, 39–43, 43]. In addition, the adoption of cloud computing and networking paradigms, such as edge cloud offloading [2, 13, 15, 16, 18, 45, 58], in-network content caching [4, 14, 48, 60, 62, 65] and device cloning [1, 3, 15, 28], has played a key role in overcoming these high bandwidth demand challenges by limiting network traffic, and in improving responsiveness of newly emerging IoT applications by reducing end-to-end latency. These newly adopted wireless technologies are very essential to empowering urban data communication infrastructures with the ability to serve, provide Internet connectivity to, and meet the quality of service requirements of millions of city users.

Recently, cloud radio access networks (C-RANs) were introduced to 5G wireless networks as a key solution to addressing capacity and coverage issues with reasonable operational costs of the network infrastructure [68]. A typical C-RAN architecture, shown in Fig. 3.1, consists of a set of light-weighted Remote Radio Heads (RRHs) covering the cell area to provide network connectivity to end users, and a base station (BS)—aka Baseband Unit or BBU—that provides connectivity to all the RRHs. All BSs are, in turn, connected to a Cloud Processing Unit (CPU) typically wired via fronthaul links. In C-RAN, the computational and storage capabilities are distributed across all the RRHs, BSs, and CPU entities, with varying capacities. These distributed storage capabilities can, therefore, be leveraged to enable in-network caching to improve network performances by reducing content access latency and network traffic congestions.

Recent studies [12] show that nearly 80% of the Internet content requested by end

users is intended only for about 1% of the content items, and about 70% of the items are being accessed only once. This content access trend can, therefore, be exploited to provide efficient in-network caching to bring frequently accessed content items closer to end users, which can lead to significant reduction of both access latency and network traffic. When designing in-network caching approaches, two typical questions need to be answered: (i) which content items need to be brought and cached in the C-RAN? and (ii) which network entities (i.e., RRH, BS, and/or CPU) are to host these content items; i.e. where should these items be placed? For addressing the first question, the consensus among recent works is to leverage recent advancements in machine/deep learning to predict which content is most likely to be accessed by end users and use such predictions to decide which items are to be cached. As for the problem of determining the right network entity for caching content items, optimal solution approaches for such an assignment problem are not practical due to the known NP-Completeness of the problem at hand, and hence, heuristic alternatives are often proposed instead, which compromise some optimality for better feasibility [56, 59, 67].

In this theis, we propose `HybridCache`, a clustered, hybrid proactive and reactive in-network caching scheme that reduces content access delay and backhaul traffic substantially. The novelty of `HybridCache` lies mainly in its ability to reduce in-network cached-item redundancy, and does so by first grouping BSs in clusters to minimize intra-cluster downloading latency and then cooperating among BSs to ensure non-redundant content placement within the same cluster. `HybridCache` ensures that the BSs belonging to the same cluster store unique content items; however, same items can still be cached in more than one cluster. We evaluate the effectiveness of `HybridCache` using the LinkNYC network as the backbone infrastructure [17, 61]. Our evaluation yields promising results and shows that `HybridCache` requires lower numbers of traversed hops and achieves higher cache hit rates, thereby reducing network content access latency and backhaul traffic. The main components of `HybridCache` are:

- K-means based clustering module that groups BSs based on their pair-wise content access delays.
- Long Short Term Memory (LSTM) based predictors that predict content popularity at each BS. We train and test the model using one-month worth of real YouTube data [44].

- A proactive caching strategy that caches and places content during off-peak hours with the aim of minimizing in-network content redundancy.
- A reactive caching strategy that updates in-network cached items upon request based on the latest content popularity.

The rest of the thesis is organized as follows. In Chapter 2, we discuss existing works on proactive & reactive caching. System model & architecture are presented in Chapter 3. Overview of proposed scheme is depicted in Chapter 4. The deep learning model and content placement schemes are introduced in Chapter 5 and 6, respectively. The proposed solutions are then analyzed and validated using LinkNYC in Chapter 7. Finally, the thesis is concluded in Chapter 8.

Chapter 2: Relater Work

Recently, some prior works on content caching in distributed mobile communication networks were proposed to cache content at network edges (e.g., BSs) [11, 59]. In [11], the authors proposed proactive caching at the BSs based on the correlation among the users and file patterns, where content is cached during off-peak hours. In [59], the authors considered the case that a user can access multiple BS caches, and formulated the cache placement as a distributed cache assignment problem and proposed an approximation algorithm for solving it. The work in [10] utilized user preference profile in a BS to derive RAN-aware proactive and reactive caching policies.

The works presented in [56,67] addressed the problem of proactive cache distribution and reactive cache replacement by formulating the problem as a class of maximizing a monotone, submodular function over a matroid constraints. In [56], the authors proposed greedy cache placement and back-tracking cache replacement algorithms, and in [67], a reactive caching scheme is elaborated by calculating the network delay reduction function when there is a request miss in the BS cache. In [48], a Deep-Cache algorithm is proposed, in which the authors predict the popularity of the content and generate fake requests out of most popular content items and they use variants of traditional reactive caching techniques, such as Least Recently Used (LRU) and Least Frequently Used (LFU) caching schemes, to cache the popular contents at the BSs.

In this work, we are exploring optimization of cache storage at BSs by clustering the nearby BSs together by minimizing the redundancy of the content items at the BS storage. The improvements in the cache storage helps maximize hit count in the network. We also propose modified variants of proactive and reactive cache schemes of previous works [56,59,67] to address the content placement problem with maximizing a monotone submodular function over a matroid constraints with sub-optimal solutions.

Chapter 3: System Model and Architecture

3.1 Cloud Radio Access Network (C-RAN) Model

We consider a general C-RAN architecture, as depicted in Fig. 3.1, whose components are: Base Band Units (BBUs), Remote Radio Heads (RRHs), and Links (Fronthaul and Backhaul connections) [67]. The users are connected (via wired or wireless) to the nearest RRHs and the RRHs are connected to BBUs. In the rest of the thesis, BBUs are also referred to as base stations (BSs). All the BSs are connected to a Cloud Processing Unit (CPU), via wired networks known as fronthaul connections, and the CPU is connected to remote servers via a high bandwidth backhaul connection. Backhaul connections are typically several times faster than fronthaul connections. The RRHs and BSs are equipped with cache storage. The BSs are also enabled with computing capabilities. The storage capacity of the CPU is typically several times higher than that of a BS, and the RRH storage is typically lesser than the BS storage.

In this system, we consider that the Central Cache Manager (CCM) is deployed at CPU, and is responsible for making the cache placement decisions using the proposed, hybrid proactive and reactive caching technique, referred to as `HybridCache` (to be described in later chapters) and for maintaining the routing tables for the content in the network. CPU receives content popularity information from BSs frequently enough to enable reactive cache replacement. RRHs are not used for proactive and reactive caching, because of their limited cache and computation capabilities. The content to be cached as predicted by the proactive caching module of `HybridCache` is fetched from the remote server during off-peak hours, so that users are not affected by the congestion caused by such proactive caching. Whenever content replacement happens at any BS cache, the routing table at the CPU is updated. Throughout this work, the size of each content item is assumed to be fixed; this assumption is mainly used for notational convenience, and can easily be lifted by splitting longer files into small blocks of same length (packetization).

The dotted circles in Fig. 3.1 represent the BS clusters, formed based on the transmis-

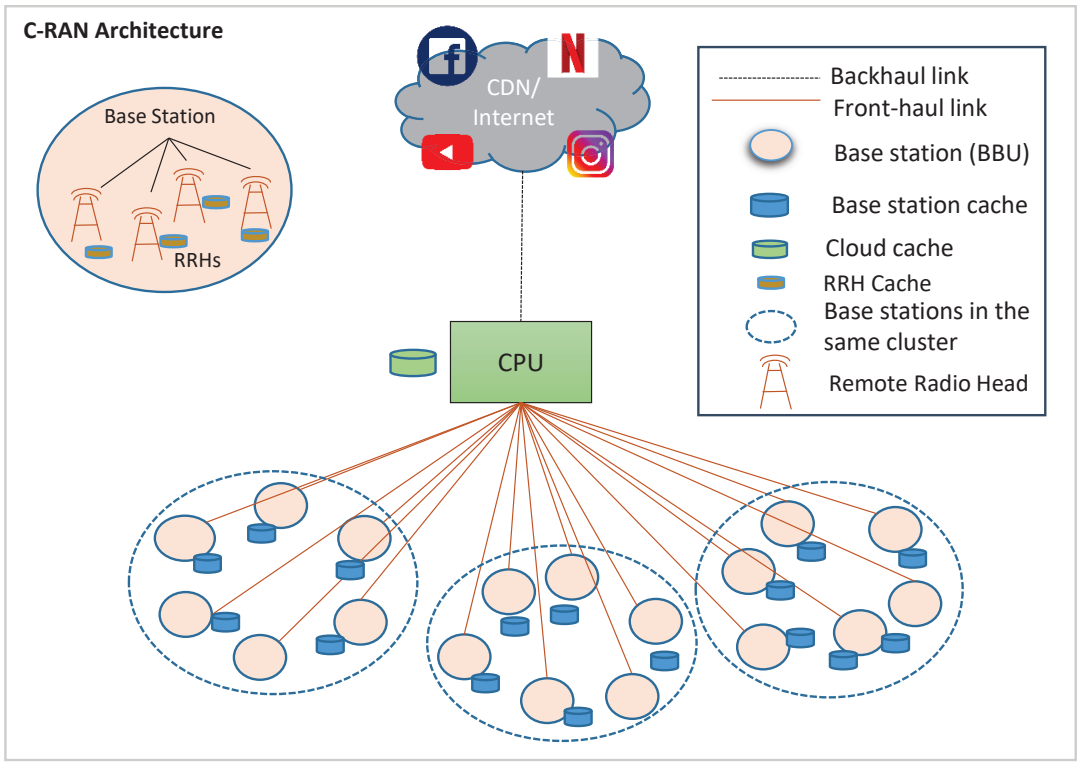


Figure 3.1: C-RAN caching system.

sion delays between the BSs. The clustering process of the BSs is described in Chapter 6.

3.2 LinkNYC Network Infrastructure

In November 2014, LinkNYC announced a project that aimed to replace all the payphones with kiosk-like structures, called Links, in the five boroughs (Manhattan, Queens, Brooklyn, Bronx and Statn Island) of New York City to provide a super-fast Internet connection to the public. One version of the network with download speeds of 280 Mbps and upload speeds of 317 Mbps is already in use. Once completed, LinkNYC will be the world's biggest and fastest public network [17].

`HybridCache` is evaluated using LinkNYC's urban communication network. The potential number of BSs in each borough depends on the number of payphones currently installed as summarized in Table 3.1 [61]. The geographic locations of payphones are shown in Fig. 3.2 with Manhattan being the most dense of all boroughs with 3409 payphones and an average distance of 43 meters. Staten Islands is the sparsest with 51 payphones and an average distance of 606 meters. In order to leverage C-RAN architecture with urban communication networks such as LinkNYC infrastructure, certain BSs/Links (i.e., payphones) are selected to serve as in-network cloud caches, and the selection is done using the hierarchical, population density based clustering heuristic proposed in [61]. The number of BSs assigned to each borough is shown in Table 3.1 with Manhattan being assigned 50 BSs and Staten Island being assigned 10 BSs.

Table 3.1
PAYPHONES IN THE FIVE BOROUGHS OF NYC [61]

Borough	# Payphones	Avg. distance	BSs
Manhattan	3409	42.3m	50
Queens	1042	136.8m	25
Brooklyn	1004	150.8m	25
Bronx	591	125.5m	20
Staten Island	51	606m	10
Total	6097	212.5m	130

The distance information between BSs is obtained from Google maps API by providing payphone location (longitude, latitude) information in selected boroughs. As done in [61], the number of hops in the transmission path between two BSs is identified by

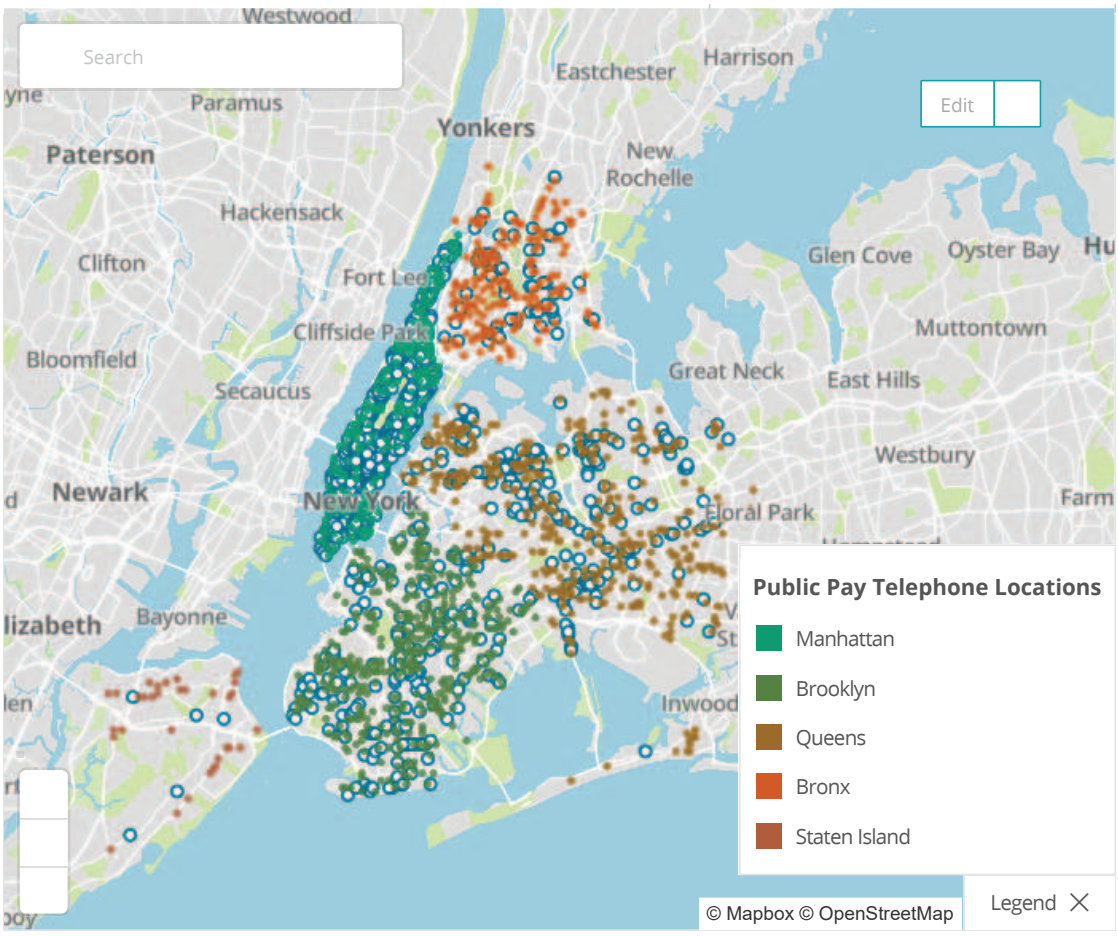


Figure 3.2: Geographic locations of payphones in NYC

first constructing the minimal spanning tree of the BSs, and then finding the shortest path between the pairs of BSs using Prim's shorted path algorithm [19].

Chapter 4: Main Components/Operations of `HybridCache`

Every time a user requests a content item from the network, the following steps are taken by `HybridCache`:

- The request is first sent to the nearest RRH and if the requested content item is present locally at the RRH, it is directly sent to the user (Case 1).
- If the content is not cached locally at the RRH, the request is forwarded to the BS to which the RRH is connected, and if the requested content is stored at the BS, it is sent to the user. This incurs lesser transmission delay when compared to fetching it from remote servers (Case 2).
- If Case 2 fails; content is not found at the BS, the request is forwarded to the CPU and if the content is found at the CPU, it is sent to the user (Case 3).
- If Case 3 fails; i.e., content is not found at CPU, the CPU searches for the content in the BSs belonging to cluster to which the BS requesting the content belongs to. If not found, it is then searched for in BSs belonging to other clusters (Case 4).
- If the content is not found in any of the BSs, the content is then fetched from the remote server (Case 5).

As shown in Fig. 4.1, the functions of `HybridCache` can be divided into *proactive cache distribution* and *reactive cache replacement*, each shown as a separate block in the figure. The proactive cache distribution function can be performed in three steps:

- **LSTM prediction**, where each BS deploys deep learning based encoder-decoder LSTM popularity predictor to forecast future content requests based on history of requests received. Popularity predictions made by LSTM predictors are sent to CPU to be used for caching placement decisions.
- **Clustering**, where the BSs are grouped into clusters based on content access delays that the BSs experience between one another. This clustering process takes place at CPU.

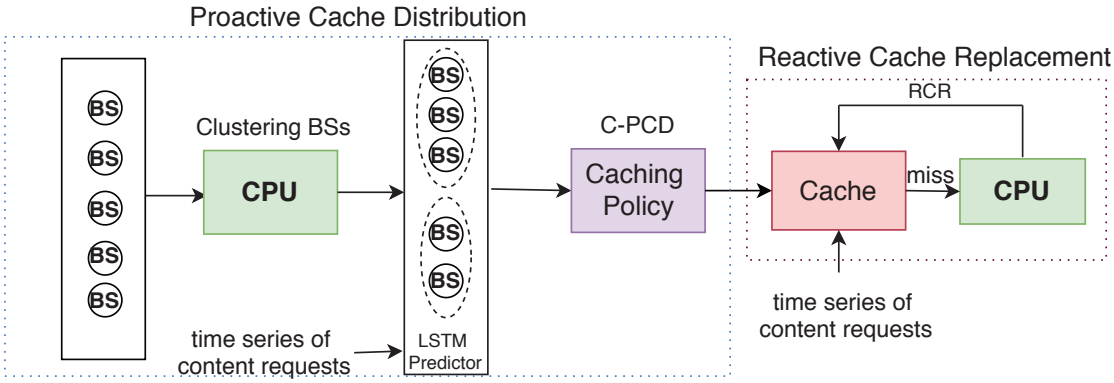


Figure 4.1: Different components/modules of `HybridCache`

- **Clustered proactive cache distribution (C-PCD)**, which is responsible for distributing content items among all BSs based on content popularity distribution at the BSs. C-PCD is performed by the CPU during off-peak hours when there is less congestion in the network.

Every time a content request comes and the requested content is not available in the local BS, the reactive cache replacement (RCR) component is triggered and run by the CPU to decide whether and where to cache the requested item in the local BS. When there is a cache miss in the local BS, the popularity information of the requested content along with all cached items are sent to CPU, and CPU invokes the RCR to decide which item to evict in the event that a decision to cache the requested item is made.

Chapter 5: Content Popularity Prediction

Each BS predicts the popularity of each of the n content items, and sends it to CPU. The CPU, upon receiving this information from all BS, uses it to run the proactive and reactive caching algorithm to determine which content to store and at which BS. For each content item, the predictor relies on past request history observed in the last k time steps to forecast requests to be received in the d future steps. Specifically, letting $x_t^{(i)}$ be the number of requests that content item i has received during time step t, the predictor takes $x_1^{(i)}, x_2^{(i)}, .., x_k^{(i)}$ as input and produces $x_{k+1}^{(i)}, x_{k+2}^{(i)}, .., x_{k+d}^{(i)}$ as output. In this work, the popularity $p_t^{(i)}$ of content item i at time step t is defined as $p_t^{(i)} = x_t^{(i)} / \sum_{j=0}^{n} x_t^{(j)}$.

5.1 Long Short Term Memory (LSTM)

Our scheme uses LSTM (Long Short Term Memory) encoder-decoder model, whose architecture is depicted in Fig. 5.1, for prediction, which is well known for achieving excellent performance in sequence to sequence predictions [37,64]. LSTM encoder-decoder allows to predict several properties of the content items, including: i) their popularity over several future time steps and ii) any sequential patterns among the requested items. Our scheme employs RNN to capture any short term and long term dependencies while predicting content popularity. It gives more flexibility in terms of predicting a variety of outputs together with varying input/output sequence length. LTSM model is quite popular due to its special design properties such as vanishing and exploding gradient

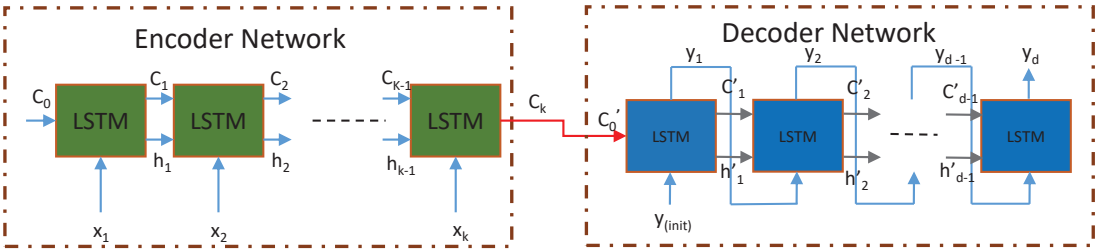


Figure 5.1: LSTM Encoder-Decoder Architecture

problem when building the deep-layer neural networks [37]. LSTM consists of cell memory to store a summary of the previous input sequence and gating mechanism to control information from input, output and cell memory. The key equations to update the LSTM weights are described below [57]:

$$\mathbf{f}_k = \sigma(W_{xf}\mathbf{x}_k + W_{hf}\mathbf{h}_{k-1} + \mathbf{b}_f) \tag{5.1}$$

$$\mathbf{i}_k = \sigma(W_{xi}\mathbf{x}_k + W_{hi}\mathbf{h}_{k-1} + \mathbf{b}_i) \tag{5.2}$$

$$\mathbf{o}_k = \sigma(W_{xo}\mathbf{x}_k + W_{ho}\mathbf{h}_{k-1} + \mathbf{b}_o) \tag{5.3}$$

$$\mathbf{c}_k = \mathbf{f}_k \odot \mathbf{c}_{k-1} + \mathbf{i}_k \odot tanh(W_{xc}\mathbf{x}_k + W_{hc}\mathbf{h}_{k-1} + \mathbf{b}_c) \tag{5.4}$$

$$\mathbf{h}_k = \mathbf{o}_k \odot tanh(\mathbf{c}_k) \tag{5.5}$$

where $\sigma(x) = 1/1 + e^{-x}$ is the sigmoid function, x_k is the kth element in the input sequence, $x \odot y$ is element wise product, $W_{xi}, W_{hi}, W_{xf}, W_{hf}, W_{xo}, W_{ho}, W_{xc}, W_{hc}$ are linear transformation matrices, $\mathbf{b}_i, \mathbf{b}_f, \mathbf{b}_o, \mathbf{b}_c$ are the bias vectors, $\mathbf{i}_k, \mathbf{f}_k, \mathbf{o}_k$ are the gating vectors, $\mathbf{c}_k$ is the cell memory state vector, and $\mathbf{h}_k$ is the state output vector. The gating vectors Eqs. (5.1), (5.2) and (5.3) determine the amount of information for the memory to update, forget and output. The cell states and outputs are updated according to Eqs. (5.4) and (5.5).

5.2 LSTM Encoder-Decoder Architecture

LSTM architecture, shown in Fig. 5.1 consists of encoder and decoder modules. The encoder takes the sequence of requests in the past as input and produces the future item requests based of temporal structure captured by encoder module. The encoder and decoder modules are trained with history of requests received for all the items, and capture any long term & short term patterns in the data. These trained modules are content popularity predictors at the BSs. The encoder module takes a k-length sequence of input values $\{x_1, x_2, .., x_k\}$ which represents the past input sequence, along with initial cell state vector, c_0. After k recursive updates of the weights via Eqs. (5.1) - (5.5), the encoder summarizes input values into the final cell state as c_k. The encoder module passes the final cell state to the decoder model, and the decoder module recursively

generates the output sequence of length d ($\{y_1, y_2, .., y_d\}$) by taking a cell state and a dummy initial decoding value $y_{(init)}$. In every update, the decoder takes the previous output value y_{d-1} as input for the current update y_d.

5.3 YouTube Dataset

The dataset [44] is retrieved from YouTube API, and has an hourly observation count (views, comments, likes, dislikes) during May 2018. These videos were released on April 2018 on YouTube. The video list was retrieved on 7th May 2018, which is the starting date of this data set. The YouTube dataset is preprocessed to evaluate proposed scheme on LinkNYC topology. The unnecessary fields are removed, and split into different sets to represent user access patterns for different BSs in Bronx and Staten Island boroughs. In total, the dataset contains 1611 content items. Each item has an entry representing the number of requests that content item received in a specified window. In our case, we consider the window size as on minute, and in total, the dataset has $41,760$ time-sequence data for each content item.

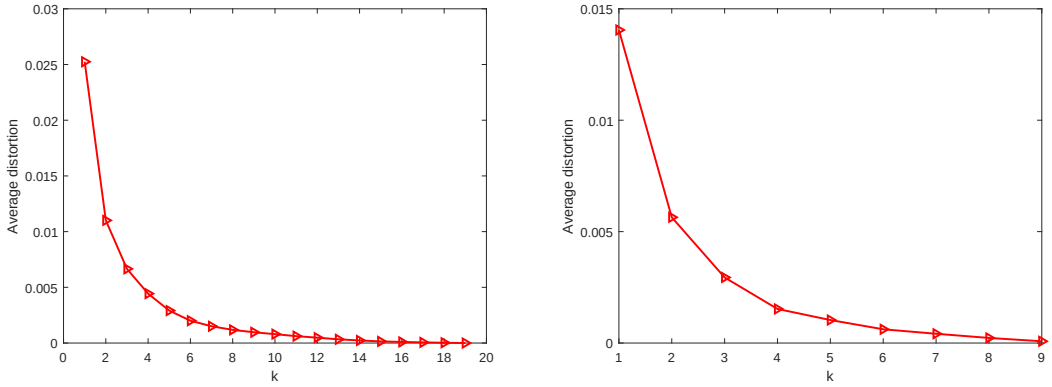
Chapter 6: Content Placement and Caching

LSTM model described in the previous chapter predicts the future content popularities and stores them at the BS to achieve efficient caching performance in individual BSs. However, because all the BSs are connected to each other via a wired connection, user-requested content can be retrieved from the neighboring BSs (if not available locally) instead of being fetched from remote servers. This achieves better overall performance since inter-BS communication is less costly than BS-to-remote sever communication.

When a content item is popular in more than one BS, it is cached in multiple BS caches (cache redundancy). This has negative impact of overall network performance. There is also a need to minimize content redundancy in BS caches for better cache utilization. If the BSs can only store unique (unredundant) content items, content requests may have to travel multiple BSs to reach the source BS, thereby creating fronthaul congestion. Therefore, the number of hops in the travel path of the requests needs to be minimized. For this reason, we propose to cluster the BSs based on transmission delay between them and store only unique contents in each clusters. Since a content item can be popular in more than one BS, optimal cache placements are known to NP-complete problems [59]. In fact, the cache placement optimization problem can be written as the maximization of a submodular function subject to matroid constraints [59].

In the proposed scheme, all the BSs in the system are clustered initially and content items are placed in the caches based on the clustered proactive cache distribution (C-PCD) algorithm described in Chapter 6.2. The data flow in the clustering & cache placement is depicted in Fig. 4.1. When there is a cache miss at BS, the reactive cache replacement (RCR) algorithm, described in Chapter 6.3, is triggered.

In the rest of the thesis, we use the following notations: The set of all BSs is denoted by $\{b_1, b_2, b_3, .., b_B\}$, where B is the number of BSs. The cache capacity is the same for all BSs and is denoted by M. The number of clusters of BSs is denoted by K. We denote by F the set of all the n content items.



(a) (b)

Figure 6.1: Selecting K values

6.1 Clustering

We use K-means algorithm [49] for clustering the BSs. Before applying k-means clustering on the BSs, the transmission delay values of the BSs are represented in two dimensional space using multidimensional scaling method [66], where the number of clusters K is determined using the elbow method [49]. With the elbow method, the largest change in the distortion determines the optimal value for K where the distortion is the sum of the distances from cluster centers of the respective clusters. In this work, we use the Euclidean distance as the distance metric.

The best value for the number of clusters K is determined by elbow method, and is shown in Figs. 6.1a and 6.1b for Bronx and Staten Islands, respectively. The largest change in the distortion occurred from 3 to 4 in Fig. 6.1a and from 2 to 3 in Fig. 6.1b, so the optimal K values are determined as 3 and 2 in Bronx and Staten Island, respectively.

Once the number of clusters K is determined, k-means clusters the BSs into K groups as follows:

1. Randomly selects K BSs, with each BS representing a point in an 2-dimensional space. These points represent initial group centroids.

2. Assigns all other BSs to their closest cluster center according to the Euclidean distance.

3. Reassigns the centroid position as mean of all BS locations in each cluster.

4. Repeat steps 2, 3 and 4 until the same points are assigned to each cluster in consecutive rounds.

6.2 Clustered Proactive Cache Distribution (C-PCD)

Let P be the popularity matrix representing the predictions made by the LSTM predictors at each BS. That is, $P = [p_{ij}]_{1 \leq i \leq B, 1 \leq j \leq n}$ where p_{ij} represents the popularity of content item j as seen by BS i. Here, p_{ij} is be used as the likelihood that content item j is to be requested by BS i. Let $X = [x_{ij}]$ be the $B \times n$ binary matrix, with x_{ij} taking 1 when item j is cached in BS i and 0 otherwise. The cache placement problem is then to find the assignment x_{ij} that

$$\begin{aligned} \text{minimize} \quad & OB(X) = \sum_{j=1}^{B} \sum_{i=1}^{n} \bar{\omega}_{ij} p_{ij} (1 - x_{ij}) \\ \text{subject to} \quad & \sum_{i=1}^{n} x_{ij} \leq M, \quad 1 \leq j \leq B \end{aligned} \tag{6.1}$$

where $\bar{\omega}_{ij}$ is the content access delay required to retrieve content item i from BS j and M is the cache capacity (in number of items) of each BS. For fixed w_{ij} and p_{ij} values, the optimization formulation in Eq. (6.1) is equivalent to

$$\begin{aligned} \text{maximize} \quad & OB'(X) = \sum_{j=1}^{B} \sum_{i=1}^{n} \bar{\omega}_{ij} p_{ij} x_{ij} \\ \text{subject to} \quad & \sum_{i=1}^{n} x_{ij} \leq M, \quad 1 \leq j \leq B \end{aligned} \tag{6.2}$$

Note that here it is realistic to assume that p_{ij} remain unchanged for sufficiently long enough time as compared to the time period during which the proactive caching is invoked and hence the optimization problem is solved. Typically, this proactive caching is invoked once a day, during off-peak time, and hence, these predicted p_{ij} values are updated accordingly.

Algorithm 1 C-PCD

Input: B,M,K

Output: $c_j \leftarrow$ filled with optimal content, $\forall j \; in \; \{1, 2, .., B, cloud\}$

for $k = 1, 2, ..., K$ **do**
  B' $\leftarrow$ number of BSs in cluster k
  $c_b \leftarrow \emptyset, \forall b \in \{1, 2, .., B'\}$
  $X \leftarrow 0$
  $count \leftarrow 0$
  while $count < M \times B'$ **do**
    $(i_o, j_o) \leftarrow$ outcome of Algorithm 2
    if c_{j_o} is full **then**
      set $x_{i_o j_o}$ to 1 in X
    else
      $c_{j_o} \leftarrow c_{j_o} + i_o$
      for $p \leq B'$ **do**
        $x_{i_o p} \leftarrow 1$
      end for
      $count \leftarrow count + 1$
    end if
  end while
  $count \leftarrow 0$
  while $count < 4M$ **do**
    $(i_o, j_o) \leftarrow$ outcome of Algorithm 2
    $c_{cloud} \leftarrow c_{cloud} + i_o$
    for $p \leq B'$ **do**
      $x_{i_o p} \leftarrow 1$
    end for
    $count \leftarrow count + 1$
  end while
end for

Algorithm 2 Calculating Marginal Value (MV).

Input: X^{prev}, B', n, OB'
Output: $(i_o j_o)$
1: set $X = 0$
2: $(i_o, j_o) = \emptyset$
3: **for** $i = 1, 2, ..., B'$ **do**
4: **for** $j = 1, 2, .., n$ **do**
5: **if** $x_{ij}! = 1$ **then**
6: $X^{curr} \leftarrow$ set $x_{ij} = 1$ in X^{prev}
7: MV_{ij}=$OB'(X^{curr}) - OB'(X^{prev})$
8: **if** $MV_{ij} > MV_{i_o j_o}$ **then**
9: $(i_o, j_o) = (i, j)$
10: **end if**
11: **end if**
12: **end for**
13: **end for**
14: Return (i_o, j_o)

`HybridCache` uses a heuristic greedy approach [59] for solving the assignment optimization problem, as shown in Algorithm 1. The algorithm determines which and where to place content items in all the BSs. Step 1 is to go through all the clusters one by one, from line 2 to 5, to initialize all required variables. At line 6, the loop is initialized to run the process until all the BS caches are full. At line 7, the entry with the maximum marginal value is selected using Algorithm 2. The entry ij from the placement matrix X with the maximal marginal value (MV_{ij}), which represents the objective function gain obtained when content item i, is cached in BS j during this step. The steps for calculating MV_{ij} value for each entry in the placement matrix X are provided in Algorithm 2. Here, B' is the number of BSs in the cluster to which the BS belongs. The algorithm returns the entry (i_o, j_o) in the placement matrix X that corresponds to the highest marginal value. The condition at line 8 is to check whether the storage is full, and if so, only that particular entry from the placement matrix X is set to 1; otherwise the content is added to BS j_o. Lines 13 to 14 are to set all entries $\{(i_0, 1), (i_o, 2), .., (i_o, B')\}$ in matrix X to 1, so that they are excluded in subsequent marginal value calculations. From line 19 to 26 is to fill the next $4M$ entries with the maximum marginal values at cloud cache.

6.3 Reactive Cache Replacement (RCR)

In the proposed scheme, the RCR procedure [56] is triggered at CPU when the requested content is not available at the local BS. The RCR scheme decides which content to evict from the BS cache to place the new content. Throughout b_{local} is a BS at which content request arrived and is fulfilled by one of the other BSs. Let C be the network delay matrix such as $C = [c_{ij}]_{1 \leq i,j \leq B}$. Each element in the matrix C for example c_{ij} refers to the transmission delay between BS i and j. The content request will be routed to the source BS with minimum transmission delay. The source BS containing content item i with the least transmission delay to BS b_{local} is $R(b_{local}, i)$ and can be found as

$$R(b_{local}, i) = \underset{b \in \{1,2,..,B\}: x_{bi}=1}{\operatorname{argmin}} c_{b_{local}b} \tag{6.3}$$

where $c_{b_{local}b}$ is transmission delay between b_{local} and b, and $x_{bi} = 1$ if the content is present in BS b and 0 otherwise. The source BS is computed at the CPU and the routing table is updated whenever there is a change in the cache storage at any BS.

When a requested content i_{new} is fetched from one of the other BSs in a cluster, the RCR is triggered and the prediction values of i_{new} and all existing items in BS b_{local} are sent to the CPU to decide whether to cache i_{new} by evicting one of the existing items i_{old} in the cache. To make that decision about cache replacement, CPU calculates the network delay reduction (NDR) for all the existing items in BS b_{local} and evicts the one with the maximum NDR to cache the new one. If all NDR values are negative, then RCR scheme does nothing. Here, NDR is defined as the reduction of average access latency in the network and can be written as:

$$\begin{aligned} NDR(b_{local}, i_{old}, i_{new}) = Cache(b_{local}, i_{new}) p(b_{local}, i_{new}) \\ - Evict(b_{local}, i_{old}) p(b_{local}, i_{old}) \end{aligned} \tag{6.4}$$

where $p(b_{local}, i_{new})$ and $p(b_{local}, i_{old})$ are the predicted popularities at BS b_{local} for item i_{new} and i_{old}, respectively. $Cache(b_{local}, i_{new})$ is the network delay reduction of caching i_{new} at BS b_{local}, which can be computed as:

$$Cache(b_{local}, i_{new}) = \sum_{b'=1}^{B} [C_{b'R(b',i_{new})} - C_{b'b_{local}}]^{+} \tag{6.5}$$

where $[x]^+$=max(x,0). The equation calculates transmission delay difference between BS to retrieve content i_{new} and what is the transmission delay if it get stored in b_{local}. We can see that if the network delay between b' and b_{local} is lower than that between b' and $R(b', i_{new})$, then the network delay is reduced by caching file i_{new}. Similarly, $Evict(b_{local}, i_{old})$ in Equation (6.4) represents the network delay of evicting i_c from the BS b_{local}, which can be computed as:

$$Evict(b_{local}, i_{old}) = \sum_{b'=1}^{B} [\min_{b \in \{a \neq b_{local} | x_{a i_{old}} = 1\}} C_{b'b} - C_{b' b_{local}}]^+ \tag{6.6}$$

Equation (6.6) computes the additional delay incurred at each BS due to eviction of content item i_{old} from BS b_{local}. We can observe that when the BS b_{local} is the source for item i_{old} for BS b', the network delay will increase if item i_{old} is evicted from BS b_{local}. The reactive caching algorithm is described in Algorithm 3, which takes b_{local}, i_{new}, cache capacity M, the network delay matrix, Routing tables and content popularity matrix as input and returns the ID of the content item in BS b_{local} with the highest DRV. If all the computed network delay reduction values are negative, then no item will be returned, meaning that no content items will be replaced with i_{new}.

Algorithm 3 Reactive Cache Replacement (RCR)

Input: b_{local}, i_{new}, M, X, C, Routing Table, P.
Output: Content item ID to replace; can be $\emptyset$

```
1: MaxValue=0
2: EvictContent=0
3: C ← Cache(b_local, i_new) returned by Eq. (6.5)
4: for i_old = 1, 2, ..., M do
5:    E ← Evict(b_local, i_old) returned by Eq. (6.6)
6:    NDR ← C − E
7:    if NDR > MaxValue then
8:       MaxValue = NDR
9:       EvictContent=i_old
10:   end if
11: end for
12: Return EvictContent
```

In Algorithm 3, the variables are initialized in lines 1 to 3. Lines 4 to 11 are used to

calculate the NDR values for all items in BS b_{local} and update the variable $EvictContent$ with the item ID that gives the maximum delay reduction value (Equation 6.4). Once Algorithm 3 completes its execution at CPU, BS b_{local} replaces item i_{new} with the content item returned by the algorithm.

Chapter 7: Performance Analysis

In this chapter, we evaluate the proposed `HybridCache` scheme on the LinkNYC's communication network. We focus on two boroughs of the LinkNYC network, namely, Bronx and Staten Island. As mentioned in Table 3.1, 20 BSs and 10 BSs are selected for Bronx and Staten Island respectively, using the heuristic proposed in [61] based on population density and the number of payphones; the BS selection process is out of the scope of this work. The download access delay between any two BSs $(BS1, BS2)$ can be expressed as:

$$delay(BS1, BS2) = \Delta.H + \Gamma.D \tag{7.1}$$

where H is the number of hops in the transmission path between $(SB1, SB2)$, D is the sum of distances between all the hops in the transmission path between $(SB1, SB2)$, Δ is processing time at each BS and Γ is transmission delay per unit distance. In our simulations, we assume Δ is ten times greater than Γ. We assume that the storage capacity of each BS is the same in the whole system. The storage capacity of the cloud cache is four times higher than BS. All BSs in the system receive content requests according to the dataset specified in Chapter 5.3. The reactive cache scheme described in Algorithm 3 is applied at every time step. Each BS deploys its own LSTM predictor for content popularity prediction, and sends this prediction information to CPU whenever there is a cache miss.

7.1 Performance Metrics and Baseline Approaches

We evaluate and show the effectiveness of `HybridCache` using three performance metrics:

(i) Cache hit ratio: the fraction of requests found locally at the BS cache.

(ii) Average access delay: the average content access latency incurred during the downloading of the content from its source to the user.

(iii) Backhaul traffic percentage (%): the fraction of traffic travelled through the back-

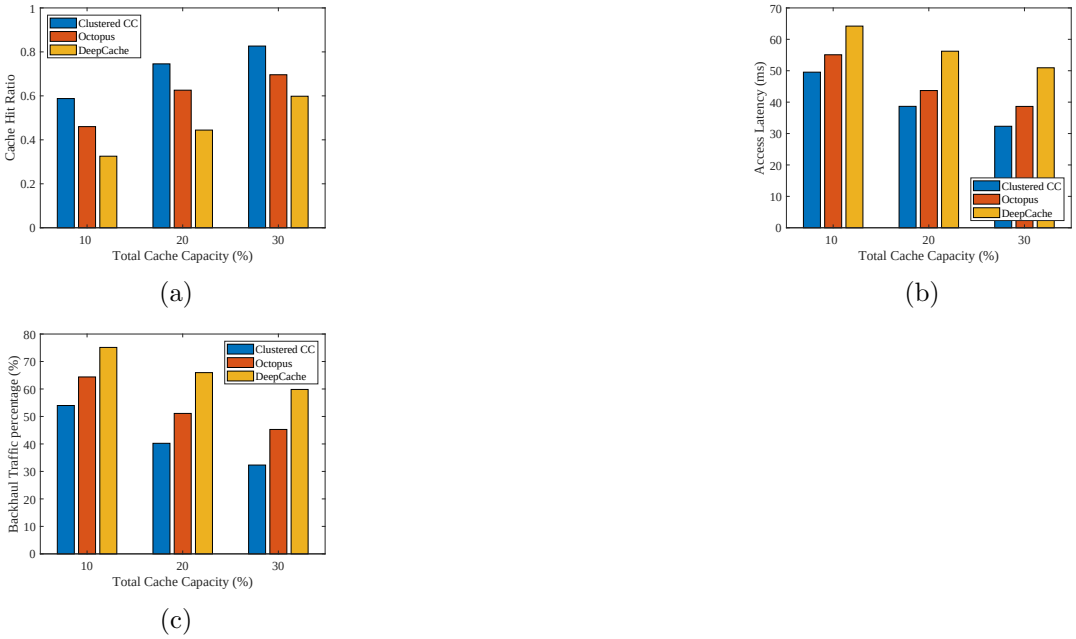
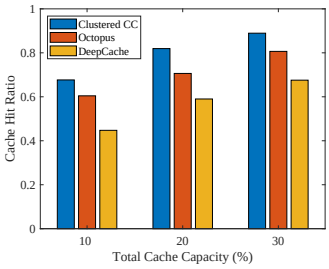
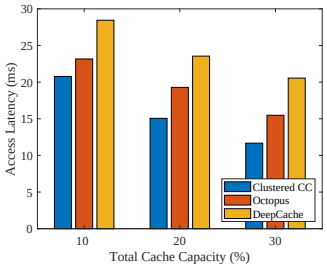


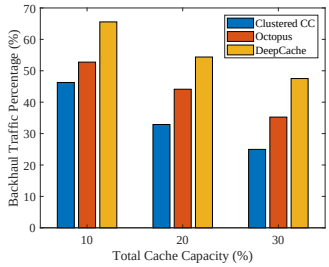
Figure 7.1: Performance evaluation of the different caching schemes using Bronx Borough of LinkNYC for K=3; the optimal number of clusters obtained by the elbow method.



(a)



(b)



(c)

Figure 7.2: Performance evaluation of the different caching schemes using Staten Island Borough of LinkNYC for K=2; the optimal number of clusters obtained by the elbow method.

haul link due to content being accessed from the remote servers.

We compare the achieved performance of `HybridCache` against two other recently proposed schemes:

Octopus [67]: In this scheme, CPU distributes the popular content among BSs during off-peak hours based on the proactive caching scheme. If the requested content is not present in any of the BSs in the system, then the content is fetched from the remote server. If the content is retrieved from a neighbouring BS, reactive cache replacement procedure is triggered and cache storage gets updated. Octopus works similar to `HybridCache` but without addressing redundant content placement.

DeepCache [48]: In this scheme, CPU distributes the popular content among the BSs based on the proactive caching scheme. If the requested content is not present in the local BS, then the content is fetched from the remote server (there is no cooperation among the BSs). At every time step, caches get adjusted based on the latest popularity values of the content items.

7.2 Result Analysis for LinkNYC Nework Topology

Fig. 7.1 (7.1a-7.1c) compares the performance of proposed `HybridCache` scheme for Bronx against Octopus and DeepCache in terms of the cache hit ratio, average access delay, and backhaul traffic load. Observe that the cache hit ratio is improved in `HybridCache` scheme over Octopus and DeepCache, because clustered BSs provide more space for the popular content items to occupy in the system by reducing in-network content redundancy. At the same time, the access delay is also reduced compared to the other two techniques because of two reasons. Firstly, clustered BSs minimize the number of hops (if content is present in one of the other BSs) to traverse the content to reach the requested user. Secondly, the BSs are clustered together based on transmission delay so there are higher chances that content is present in the BS with lesser transmission delay. The backhaul congestion is lesser because the BS storage is optimized to store greater numbers of content items by avoiding redundancy. For example, when the storage capacity in the system is 30% of all the contents, the average access delay is reduced by about 16% over Octopus and by about 36% over DeepCache. For the same storage capacity, cache hit ratios are improved by about 19% and 38% and the backhaul loads are reduced by about 28% and 46% over Octopus and DeepCache, respectively.

Fig. 7.2 (7.2a-7.2c) shows the performance evaluation using the Staten Island area of LinkNYC network. The figure also shows results comparing the performance of proposed `HybridCache` scheme against Octopus and DeepCache in terms of the cache hit ratio, average access delay, and backhaul traffic load. When the storage capacity in the system is 30% of all the contents, the average access delay is reduced by about 25% over Octopus and by about 43% over DeepCache. With the same storage capacity, cache hit ratios are improved by about 10% and 32% and the backhaul traffic loads are reduced by about 29% and 47% over Octopus and DeepCache, respectively. If we compare the results with Bronx borough, we are storing higher numbers of items in each BS, since Staten Island has lesser number of BSs than in Bronx, thereby resulting in better improvements.

Observe that average access latency is lower in State Island than in Bronx. This is because there is a lower number of BSs in State Island, which leads to lesser numbers of hops the content needs to travel to reach the end user. Since the cache capacity is the same in both boroughs, the users in State Island are more likely to fetch the content from one of the neighboring BSs in comparison to Bronx. The higher number of BSs causes varied access patterns which leads to higher number of popular content being stored in the network. At the same time, cache hit ratio is improved and backhaul load is decreased in Staten Island due to the higher number of contents available to the users from nearby BSs. In all the cache capacities for both the regions of LinkNYC, our proposed technique always performs the best in all performance metrics.

Fig.7.3 (7.3a-7.3c) shows the performance of the proposed `HybridCache` scheme under different k values with 10% of content items in the system. We can observe that when we choose the best k value in the clustering process, the performance of the proposed scheme achieves the best results. The figures here are based on Bronx area of LinkNYC network. We can observe that the optimal value is 2 in the graph in which the proposed clustered approach achieves best performance in terms of cache hit ratio, average access latency, and backhaul traffic load.

7.3 Result Analysis for Random Nework Topology

Fig.7.4 (7.4a-7.4c) shows the performance evaluation comparison of the different caching schemes but while considering random network topologies. In the random topology, the distance between BSs are chosen randomly. The prims minimal spanning tree [19] is built

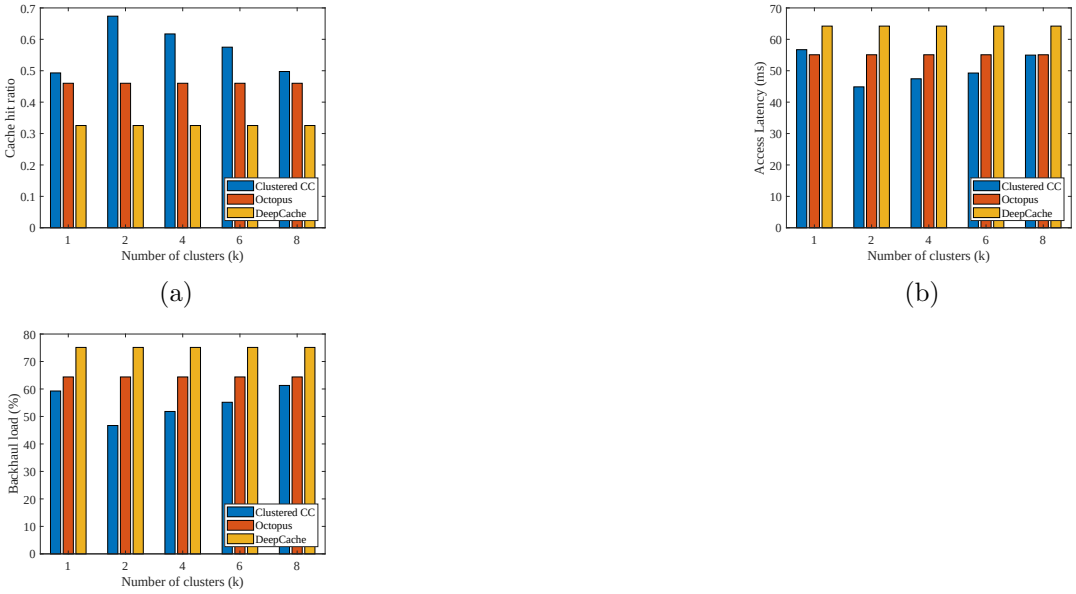


Figure 7.3: Performance evaluation under varying K values for Bronx Borough: The BS cache capacity is 10%.

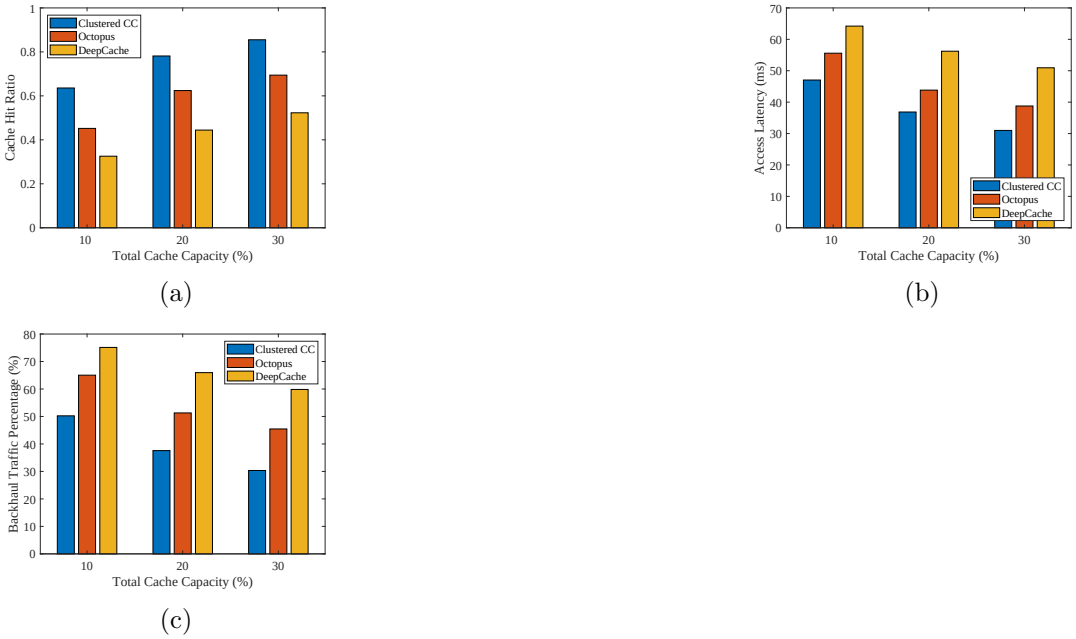


Figure 7.4: Performance evaluation of different caching schemes under random network topologies, with K=2; the optimal number of clusters as obtained by the elbow method.

upon the random topology, and the number of hops between any two BSs is computed based on prims shortest path algorithm. The transmission delay between any two BS is computed using Equation (7.1). The figure shows that our proposed `HybridCache` scheme outperforms Octopus and DeepCache schemes in terms of the hit ratio, access delay and backhaul traffic load in random topology too.

Chapter 8: Conclusion

In this thesis, we proposed `HybridCache` scheme for urban communication networks (LinkNYC) built on as Cloud Radio Access Networks (C-RANs). In the proposed scheme, all BSs are connected to CPU which manages the traffic in the network, and clustered proactive cache distribution (C-PCD) & reactive cache replacement (RCR) algorithms with LSTM content popularity predictors at BSs provide users reduces average access delay. We evaluated the proposed technique using real-time YouTube dataset on LinkNYC topology specifically the Bronx and Staten Island boroughs. It is demonstrated that the proposed `HybridCache` scheme with LSTM popularity predictors outperforms Octopus and DeepCache schemes in both the boroughs (Bronx, Staten Island) of LinkNYC topology. Simulations results reveal that with a storage capacity 30% of all contents in the system, our proposed clustered proactive caching scheme yields 15.57% improvement in the cache hit ratio, as well as 18.14% and 29.69% decrease in average access latency and backhaul traffic load respectively as compared to the Octopus scheme.

Bibliography

[1] Sherif Abdelwahab and Bechir Hamdaoui. Flocking virtual machines in quest for responsive iot cloud services. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2017.

[2] Sherif Abdelwahab, Bechir Hamdaoui, Mohsen Guizani, and Ammar Rayes. Enabling smart cloud services through remote sensing: An internet of everything enabler. *IEEE Internet of Things Journal*, 1(3):276–288, 2014.

[3] Sherif Abdelwahab, Sophia Zhang, Ashley Greenacre, Kai Ovesen, Kevin Bergman, and Bechir Hamdaoui. When clones flock near the fog. *IEEE Internet of Things Journal*, 2018.

[4] Ragda Abuhadra and Bechir Hamdaoui. Proactive in-network caching for mobile on-demand video streaming. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2018.

[5] Nadia Adem and Bechir Hamdaoui. Delay performance modeling and analysis in clustered cognitive radio networks. In *2014 IEEE Global Communications Conference*, pages 193–198. IEEE, 2014.

[6] Nadia Adem and Bechir Hamdaoui. The impact of stochastic resource availability on cognitive network performance: modeling and analysis. *Wireless Communications and Mobile Computing*, 2015.

[7] Nadia Adem and Bechir Hamdaoui. Jamming resiliency and mobility management in cognitive communication networks. In *Communications (ICC), 2017 IEEE International Conference on*, pages 1–6. IEEE, 2017.

[8] Nadia Adem, Bechir Hamdaoui, and Attila Yavuz. Pseudorandom time-hopping anti-jamming technique for mobile cognitive users. In *2015 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6. IEEE, 2015.

[9] Nadia Adem, Bechir Hamdaoui, and Attila Yavuz. Mitigating jamming attacks in mobile cognitive networks through time hopping. *Wireless Communications and Mobile Computing*, 2016.

[10] Hasti Ahlehagh and Sujit Dey. Video-aware scheduling and caching in the radio access network. *IEEE/ACM Transactions on Networking (TON)*, 22(5):1444–1462, 2014.

[11] Ejder Bastug, Mehdi Bennis, and Mérouane Debbah. Living on the edge: The role of proactive caching in 5g wireless networks. *IEEE Communications Magazine*, 52(8):82–89, 2014.

[12] Soumya Basu, Aditya Sundarrajan, Javad Ghaderi, Sanjay Shakkottai, and Ramesh Sitaraman. Adaptive ttl-based caching for content delivery. In *ACM SIGMETRICS Performance Evaluation Review*, volume 45, pages 45–46. ACM, 2017.

[13] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16. ACM, 2012.

[14] Mingzhe Chen, Walid Saad, Changchuan Yin, and Mérouane Debbah. Echo state networks for proactive caching in cloud-based radio access networks with mobile users. *IEEE Transactions on Wireless Communications*, 16(6):3520–3535, 2017.

[15] Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, and Ashwin Patti. Clonecloud: elastic execution between mobile device and cloud. In *Proceedings of the sixth conference on Computer systems*, pages 301–314. ACM, 2011.

[16] Eduardo Cuervo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, and Paramvir Bahl. Maui: making smartphones last longer with code offload. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 49–62. ACM, 2010.

[17] Devindra. Linknyc's free gigabit wifi is here, and it is glorious, Jul 2019.

[18] Jason Flinn. Cyber foraging: Bridging mobile and cloud computing. *Synthesis Lectures on Mobile and Pervasive Computing*, 7(2):1–103, 2012.

[19] John C Gower and Gavin JS Ross. Minimum spanning trees and single linkage cluster analysis. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 18(1):54–64, 1969.

[20] Mohamed Grissa, Bechir Hamdaoui, and Attila A Yavuz. Location privacy in cognitive radio networks: A survey. *IEEE Communications Surveys & Tutorials*, 2017.

[21] Mohamed Grissa, Bechir Hamdaoui, and Attila A Yavuz. Unleashing the power of multi-server pir for enabling private access to spectrum databases. *IEEE Communications Magazine*, 2018.

[22] Mohamed Grissa, Attila Yavuz, and Bechir Hamdaoui. An efficient technique for protecting location privacy of cooperative spectrum sensing users. In *Computer Communications Workshops (INFOCOM WKSHPS), 2016 IEEE Conference on*, pages 915–920. IEEE, 2016.

[23] Mohamed Grissa, Attila Yavuz, and Bechir Hamdaoui. When the hammer meets the nail: Multi-server PIR for database-driven CRN with location privacy assurance. In *Communications and Network Security (CNS), 2017 IEEE Conference on*. IEEE, 2017.

[24] Mohamed Grissa, Attila A Yavuz, and Bechir Hamdaoui. Cuckoo filter-based location-privacy preservation in database-driven cognitive radio networks. In *Computer Networks and Information Security (WSCNIS), 2015 World Symposium on*, pages 1–7. IEEE, 2015.

[25] Mohamed Grissa, Attila A Yavuz, and Bechir Hamdaoui. Location privacy preservation in database-driven wireless cognitive networks through encrypted probabilistic data structures. *IEEE Transactions on Cognitive Communications and Networking*, 3(2):255–266, 2017.

[26] Mohamed Grissa, Attila A Yavuz, and Bechir Hamdaoui. Preserving the location privacy of secondary users in cooperative spectrum sensing. *IEEE Transactions on Information Forensics and Security*, 12(2):418–431, 2017.

[27] Mohamed Grissa, Attila A Yavuz, and Bechir Hamdaoui. TrustSAS: A trustworthy spectrum access system for the 3.5 ghz cbrs band. In *INFOCOM*. IEEE, 2019.

[28] Karim Habak, Mostafa Ammar, Khaled A Harras, and Ellen Zegura. Femto clouds: Leveraging mobile devices to provide cloud service at the edge. In *2015 IEEE 8th International Conference on Cloud Computing (CLOUD)*, pages 9–16. IEEE, 2015.

[29] Bechir Hamdaoui. Adaptive spectrum assessment for opportunistic access in cognitive radio networks. *IEEE Transactions on Wireless Communications*, 8(2):922–930, 2009.

[30] Bechir Hamdaoui, Bassem Khalfi, and Mohsen Guizani. Compressed wideband spectrum sensing: Concept, challenges, and enablers. *IEEE Communications Magazine*, 56(4):136–141, 2018.

[31] Bechir Hamdaoui, MohammadJavad NoroozOliaee, Kagan Tumer, and Ammar Rayes. Aligning spectrum-user objectives for maximum inelastic-traffic reward. In *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*, pages 1–6. IEEE, 2011.

[32] Bechir Hamdaoui, MohammadJavad NoroozOliaee, Kagan Tumer, and Ammar Rayes. Coordinating secondary-user behaviors for inelastic traffic reward maximization in large-scale osa networks. *Network and Service Management, IEEE Transactions on*, 9(4):501–513, 2012.

[33] Bechir Hamdaoui and Parameswaran Ramanathan. A cross-layer admission control framework for wireless ad-hoc networks using multiple antennas. *IEEE Transactions on Wireless Communications*, 6(11), 2007.

[34] Bechir Hamdaoui and Parameswaran Ramanathan. Cross-layer optimized conditions for qos support in multi-hop wireless networks with mimo links. *IEEE Journal on Selected Areas in Communications*, 25(4), 2007.

[35] Bechir Hamdaoui and Kang G Shin. Characterization and analysis of multi-hop wireless mimo network throughput. In *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, pages 120–129. ACM, 2007.

[36] Bechir Hamdaoui and Kang G Shin. OS-MAC: An efficient mac protocol for spectrum-agile wireless networks. *IEEE Transactions on Mobile Computing*, 7(8):915–930, 2008.

[37] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[38] Cisco Visual Networking Index. Global mobile data traffic forecast update, 2016–2021 white paper. *Cisco: San Jose, CA, USA*, 2017.

[39] Bassem Khalfi, Abdurrahman Elmaghbub, and Bechir Hamdaoui. Distributed wideband sensing for faded dynamic spectrum access with changing occupancy. In *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2018.

[40] Bassem Khalfi, Bechir Hamdaoui, and Mohsen Guizani. Extracting and exploiting inherent sparsity for efficient iot support in 5G: Challenges and potential solutions. *IEEE Wireless Communications Magazine*, 24(5):68–73, 2017.

[41] Bassem Khalfi, Bechir Hamdaoui, and Mohsen Guizani. AirMAP: Scalable spectrum occupancy recovery using local low-rank matrix approximation. In *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2018.

[42] Bassem Khalfi, Bechir Hamdaoui, Mohsen Guizani, and Nizar Zorba. Efficient spectrum availability information recovery for wideband dsa networks: A weighted compressive sampling approach. *IEEE Transactions on Wireless Communications*, 17(4):2162–2172, 2018.

[43] Bassem Khalfi, Adem Zaid, and Bechir Hamdaoui. When machine learning meets compressive sampling for wideband spectrum sensing. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2017 13th International*, pages 1120–1125. IEEE, 2017.

[44] Kmmd. Youtube videos viewcount every hour, Jun 2018.

[45] Mads Darø Kristensen. Execution plans for cyber foraging. In *Proceedings of the 1st workshop on Mobile middleware: embracing the personal communication device*, page 2. ACM, 2008.

[46] Megha Maiya and Bechir Hamdaoui. An improved ieee 802.11 mac protocol for wireless ad-hoc networks with multi-channel access capabilities. In *High Performance Computing and Simulation (HPCS), 2011 International Conference on*, pages 162–168. IEEE, 2011.

[47] Megha Maiya and Bechir Hamdaoui. imac: improved medium access control for multi-channel multi-hop wireless networks. *Wireless Communications and Mobile Computing*, 13(11):1060–1071, 2013.

[48] Arvind Narayanan, Saurabh Verma, Eman Ramadan, Pariya Babaie, and Zhi-Li Zhang. Deepcache: A deep learning based framework for content caching. In *Proceedings of the 2018 Workshop on Network Meets AI & ML*, pages 48–53. ACM, 2018.

[49] Yan Niu, Shen Gao, Nan Liu, Zhiwen Pan, and Xiaohu You. Clustered small base stations for cache-enabled wireless networks. In *2017 9th International Conference on Wireless Communications and Signal Processing (WCSP)*, pages 1–6. IEEE, 2017.

[50] M NoroozOliaee, Bechir Hamdaoui, and Kagan Tumer. Achieving optimal elastic traffic rewards in dynamic multichannel access. In *High Performance Computing and Simulation (HPCS), 2011 International Conference on*, pages 155–161. IEEE, 2011.

[51] MohammadJavad NoroozOliaee and Bechir Hamdaoui. Analysis of guard-band-aware spectrum bonding and aggregation in multi-channel access cognitive radio networks. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2016.

[52] MohammadJavad NoroozOliaee, Bechir Hamdaoui, Xiuzhen Cheng, Taieb Znati, and Mohsen Guizani. Analyzing cognitive network access efficiency under limited spectrum handoff agility. *IEEE Transactions on Vehicular Technology*, 63(3):1402–1407, 2014.

[53] MohammadJavad NoroozOliaee, Bechir Hamdaoui, and Mohsen Guizani. Maximizing secondary-user satisfaction in large-scale dsa systems through distributed team cooperation. *Wireless Communications, IEEE Transactions on*, 11(10):3588–3597, 2012.

[54] MohammadJavad NoroozOliaee, Bechir Hamdaoui, and Kagan Tumer. Efficient objective functions for coordinated learning in large-scale distributed osa systems. *Mobile Computing, IEEE Transactions on*, 12(5):931–944, 2013.

[55] MohammadJavad NoroozOliaee, Bechir Hamdaoui, Taieb Znati, and Mohsen Guizani. Forced spectrum access termination probability analysis under restricted channel handoff. In *WASA*, pages 358–365. Springer, 2012.

[56] Haitian Pang, Jiangchuan Liu, Xiaoyi Fan, and Lifeng Sun. Toward smart and cooperative edge caching for 5g networks: A deep learning based approach. In *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, pages 1–6. IEEE, 2018.

[57] Seong Hyeon Park, ByeongDo Kim, Chang Mook Kang, Chung Choo Chung, and Jun Won Choi. Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1672–1678. IEEE, 2018.

[58] Mahadev Satyanarayanan, Paramvir Bahl, Ramón Caceres, and Nigel Davies. The case for vm-based cloudlets in mobile computing. *Pervasive Computing, IEEE*, 8(4):14–23, 2009.

[59] Karthikeyan Shanmugam, Negin Golrezaei, Alexandros G Dimakis, Andreas F Molisch, and Giuseppe Caire. Femtocaching: Wireless video content delivery through distributed caching helpers. *arXiv preprint arXiv:1109.4179*, 2011.

[60] Hassan Sinky and Bechir Hamdaoui. Cloudlet-aware mobile content delivery in wireless urban communication networks. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7. IEEE, 2016.

[61] Hassan Sinky, Bassem Khalfi, Bechir Hamdaoui, and Ammar Rayes. Responsive content-centric delivery in large urban communication networks: A linknyc use-case. *IEEE Transactions on Wireless Communications*, 17(3):1688–1699, 2017.

[62] Hassan Sinky, Bassem Khalfi, Bechir Hamdaoui, and Ammar Rayes. Responsive content-centric delivery in large urban communication networks: A linknyc use-case. *IEEE Transactions on Wireless Communications*, 17(3):1688–1699, 2018.

[63] Akhil Sivanantha, Bechir Hamdaoui, Mohsen Guizani, Xiuzhen Cheng, and Taieb Znati. Em-mac: An energy-aware multi-channel mac protocol for multi-hop wireless networks. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2012 8th International*, pages 1159–1164. IEEE, 2012.

[64] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[65] Yongxue Tian and Li Pan. Predicting short-term traffic flow by long short-term memory recurrent neural network. In *2015 IEEE international conference on smart city/SocialCom/SustainCom (SmartCity)*, pages 153–158. IEEE, 2015.

[66] Warren S Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17(4):401–419, 1952.

[67] Tuyen X Tran, Abolfazl Hajisami, and Dario Pompili. Cooperative hierarchical caching in 5g cloud radio access networks. *IEEE Network*, 31(4):35–41, 2017.

[68] Tuyen X Tran and Dario Pompili. Dynamic radio cooperation for user-centric cloud-ran with computing resource sharing. *IEEE Transactions on Wireless Communications*, 16(4):2379–2393, 2017.

[69] Pavithra Venkatraman and Bechir Hamdaoui. Cooperative q-learning for multiple secondary users in dynamic spectrum access. In *2011 7th International Wireless Communications and Mobile Computing Conference*, 2011.

[70] Pavithra Venkatraman, Bechir Hamdaoui, and Mohsen Guizani. Opportunistic bandwidth sharing through reinforcement learning. *Vehicular Technology, IEEE Transactions on*, 59(6):3148–3153, 2010.