Heuristic Prefetching Caching Strategy to Enhance QoE in Edge Computing

Meng Sun, Haopeng Chen, Buqing Shu
School of Electronic Information and Electronic Engineering
Shanghai Jiao Tong University
Shanghai, China
Email: {sun-meng,chen-hp,daemondshu}@sjtu.edu.cn

Fei Hu
China Aeronautical Radio Electronics Research Institute
Shanghai, China
Email: hu_fei@careri.com

***Abstract*—The emergence of mobile devices and wireless services brings unprecedented traffic demand, which causes the bad quality of experience arises in traditional reactive networks, such as long loading time and loss of responsiveness. This paper presents the heuristic prefetching caching strategy in 5G networks, which prefetches content based on its historical frequency in order to improve the quality of experience. The cache of the base station is split into the proactive one and the reactive one. The proactive cache prefetches the popular content within the limit of capacity for a sum total maximum of frequency, while the reactive one caches others with the ordinary caching algorithm. At each period, we adjust the proportion of the proactive cache to minimize latency based on the idea of Simulated Annealing.**

Under the circumstances where all the content is predictable, our caching strategy improves hit ratio by 20%. And it reduces latency by 15% in the architecture of 400MB small base stations and even 52% with 200MB small base stations, which could enhance the quality of experience to a great degree.

***Keywords*-popularity of content; prediction; heuristic prefetching caching strategy; proportion;**

I. Introduction

With the increasing popularity of mobile devices, various wireless services have yielded unprecedented traffic demands, which brought heavy burden to network. One promising approach to meet these demands is via the deployment of small cell networks, which are based on the idea of deploying short-range, low-power, and low-cost small base stations underlaying the macrocellular network [1]. Therefore, a dense deployment of wireless networks has been built especially in urban environments, including access points and cellular base stations [2].

However, traditional reactive networking paradigm, in which contents are fetched after the arrival of requests, brings a series of problems such as low hit ratio and high latency. In addition, the loading process is not smooth when requesting the large content divided into chunks because of high latency between adjacent chunks.

The popularity distribution of content is generally modeled as a ZipF distribution [3],that is, a fraction of content is frequently used. However, the popular content is usually requested alternately, which would cause cache thrashing. Actually, a few popular data items account for most of the traffic load and are requested by users at different times [4].

In this paper, we promote our content delivery strategy in order to lighten the traffic load and enhance the quality of experience(QoE) for users. In general, our caching strategy split the cache of an edge node into the proactive one and the reactive one.

The popularity-based caching strategy for the content-centric network called MPC has been proposed in the literature [5], which achieves a higher hit ratio while caching less content. To reduce the occurrence of cache thrashing, the base station is supposed to prefetch popular content into its proactive cache. Each edge node counts the number of requests to collect the popularity of each content item. Besides, if the popularity should be known in advance, we could use the knapsack arithmetic to allocate popular content each base station for the largest sum of popularity within the limit of cache capacity rather than simply tag popular content whose count reaches a popularity threshold.

Many studies [1], [2], [6], [7] have focused on prefetch strategy with the prediction of the statistical pattern [6], user demand [1], user mobility [8] or network level mobility [2]. Although human behavior is highly predictable and correlated [9], predicting the accurate time and position of user requests is not obtainable [6]. Due to complex road networking and various social events, the prediction of mobility patterns does not hold in urban areas. Under this condition, we develop a crowd behavior time-series prediction model based on aggregated information of content popularity collected by edge nodes. In other words, statistical patterns such as popularity distribution of content might enable a more accurate level of prediction than individual behavior. The long short-term memory (LSTM) is not the only neural network which learns a context-sensitive language [10]. Furthermore, each content item has the historical record of time period and frequency, which is suitable for LSTM prediction. In my previous work which is published in progress, I assume that all the content is limited to a fraction of data then the popularity of content is known a priori. The previous model is not applicable to a particular situation that there are new content whose popularity cannot be predicted. Now, we also consider new items which do not have enough historical record to train the prediction model. As for old items whose historical record is enough,

978-1-7281-1141-4/18/$31.00 ©2018 IEEE
DOI 10.1109/BDCloud.2018.00073

we calculate the R^2 to evaluate the accuracy of prediction. Then we define a threshold that classifies the prediction of old items into two categories, accurate and inaccurate. As a consequence, all the content is classified into two categories, predictable and unpredictable.

1) Predictable content which could be predicted with tiny distinguish. The R^2 of prediction is beyond the threshold.
2) Unpredictable content which includes new items and old items whose predictions are relatively inaccurate.

After the prediction of popularity, we use the predicted popularity of predictable content and the requested count of unpredictable content in the last one period as the predicted frequency of each period. At the beginning of each period, each small base station adjusts to the optimal proportion for the optimal latency based on the idea of the simulated annealing algorithm. Then each small base station prefetches popular content into its proactive cache based on the knapsack problem algorithm.

During each period, once the small base station receives requests, its proactive cache would be firstly looked up. If the requested item is prefetched before, this request would be returned directly. Otherwise, we would look up the reactive cache. If the reactive cache is also a miss, the request is transferred to the upper base station.

In this paper, we have made the following contributions:

- A simple method to collect the popularity of content from each small cell base station in the architecture of edge computing.
- A time-series prediction model to predict the popularity of content with the collected historical statistics.
- Heuristic prefetching caching strategy that partitions cache into the proactive one which prefetches popular content with predicted popularity and the reactive one where other contents are cached with the ordinary caching algorithm such as Least Recently Used(LRU).
- A cache partition algorithm to adjust to the optimal proportion of the proactive cache based on the idea of simulated annealing.

The rest of this paper is organized as follows: section II gives an overview of edge caching, popularity of content and Long Short-Term Memory with related researches. In section III, we present LSTM prediction model with our definition of content popularity and classify content into predictable and unpredictable ones. We describe the architecture of the 5G network, present heuristic prefetching caching strategy and introduce the method to configure the proportion of proactive cache in section IV. Then we describe our scenario, demonstrate our simulation, and analyze the result in section V. Section VI concludes our discussion.

II. Related Works

A. *Edge Caching*

As the demand for rich multimedia services in mobile networks soars, improving spectral efficiency for 5G cellular networks becomes important which can be achieved by enhancing the exploitation of spatial resources [4]. It is proved that the latency and throughput can be optimized by reducing the distance between base stations and users. However, this approach is not practical because of the high cost.

Reducing round trips and shortening transmission distance seem to be promising ideas. Edge Computing, characterized by proximity to end users, dense geographical distribution, and support for mobility, provides low latency, location awareness, improved QoS, and heterogeneity support [11]. Obviously, caching content at the edge would reduce the distance to deliver. The article [12] made a summary of the common approaches for content caching that have already been adopted on the Internet nowadays. As the work illustrated, moving content towards the network edge closer to users would reduce network traffic and improve quality of experience.

Researchers and engineers have been investigating effective ways to reduce the duplicate content transmissions by adopting intelligent caching strategies [13]. In literature [4], caching popular content locally before relevant requests actually arrive is more attractive with the rapid growth of storage capacity of devices. The authors of [14] exploit the coverage number, i.e. the number of base stations simultaneously covering a user, to cache the most popular content. This strategy maximizes performance with less cost of storage.

When it comes to the cache location, there are a great many choices, such as the evolved packet core, the base stations and the devices of users [4]. Compared with the evolved packet core, caching at wireless networks alleviates backhaul congestion. As for caching at users, it is feasible to cache the latest or the most popular content for oneself but not practical to cache content for other users. Thus it makes no sense to cache at users when prediction algorithms based on the crowd behavior are employed. The research [15] presents FemtoCaching and evaluates the efficiency of the video content delivery through distributed caching helpers in the femtocell network. Furthermore, many recent works have investigated that caching data in densely deployed small-cell base stations gains considerable benefits.

B. *Popularity of Content*

Researchers present different definitions of the term: popularity of content. In literature [4], popularity of content is the ratio of the number of requests for a particular piece of data to the total number of requests. It is usually computed in the context of a certain region during a given period of time. Besides, the author regards predicting the popularity of content in the coverage of a base station as a challenge because of the number of users associated with the base station is highly dynamic. However, the changeable number of users does not matter since we are only concerned about the crowd behavior. To put it more simply, we collect the total count of the requested piece of data through base stations rather than users. In another work [5], the term is defined differently. Every edge node counts the number of requests for each piece of data as popularity and select popular content with a certain threshold. However, without comparison with others, the number is meaningless.

According to the research [15], the popularity distribution of content changes at a much slower rate than the traffic variation of cellar network. The popularity distribution of content in a network is generally modeled as a ZipF distribution [3]. In other words, a fraction of content is frequently used, which provides theoretical basis for the assumption that the requested content is limited within a small portion of data thus the popularity of content is known a priori as stated in literature [14]. In my previous work, this assumption is also adopted to simplify the problem, but it is more precise and practical to take new content which has not enough historical data to train the prediction model into consideration.

C. Long Short-Term Memory

Predicting popularity of content has been extensively studied in work [3], [16], [17]. Among all of relevant algorithms, time series models are widely adopted such as autoregressive integrated moving average [18], regression models [19], and classification models [20].

However, prediction methods relying on existing trained models would be outdated and produce inaccurate results [21]. Besides, predicting with the latest data also leads to a time delay because the training process costs a long time. Long Short-Term Memory(LSTM), a specific recurrent neural network architecture, which has been successfully applied to various sequence prediction [22], is the most suitable for predicting the popularity of content in this situation. LSTM has the ability to model the short-term memory which can last for a long period of time using gating mechanisms [23]. In one word, these features of LSTM improve the accuracy of prediction and reduce or even eliminate the delay.

III. Popularity Prediction

In this section, we first give our definition to the popularity of content. The we present the method to collect corresponding information and the prediction algorithm, which provide the predicted frequency for base stations to prefetch popular data items.

Algorithm 1 procedure of LSTM prediction

```
$contentList$ = List of content
Initialize $rawValues$
**for** each $content \in contentList$ **do**
    initialize $values$ with $rawValues_{content}$;
    calculate differencing of $values$ with the previous one;
    transform $values$ to supervised value;
    scale supervised value;
    $model$ = fit lstm with scaled data;
    predict with $model$;
    invert result;
**end for**
```

A. Popularity of Content

It was mentioned that there are various methods to define the popularity of content, such as the ratio of the number of requests for a particular data item to the total number of requests and the count of requested content. Actually, they are inherently the same because the popularity of content makes sense by comparison. Therefore, we define the number of requests for the specific piece of data during a certain period as the popularity of that data item.

A table of popularity is maintained for each base during a certain period. The table, recording the name of each data item and its corresponding count, would be fed into the prediction service.

B. Time Series Prediction Model

Having the popularity of content collected, every content has a set of records in the form of the pair (time period, popularity), which are suitable for time series prediction. Compared with ordinary neural networks, LSTM is a model for the short-term memory which can last for a long period of time using gate mechanisms. It optimizes the time-consuming and inaccurate prediction to a great degree. Thus we employ LSTM to train prediction model and then predict the popularity of content in the next few periods. The procedure is shown in Algorithm 1.

The first step is to load data which contains time and number of requests over the corresponding period. However, prediction with LSTM recurrent neural networks consumes a long time so that we predict the popularity of content for a few periods. For example, at 2018-8-10 7:00, we need to predict the popularity of content at 2018-8-10 8:00, 2018-8-10 9:00, 2018-8-10 10:00 or more. We should guarantee that the popularity of content has been predicted when we need it to prefetch popular data items. If the prediction costs two hours, the predicted popularity at 9:00 can be obtained in time because we have started the prediction at 7:00. What's more, we update the prediction model every period and then apply the latest result.

For each data item, we make data stationary with differencing because stationary data is easier to model and

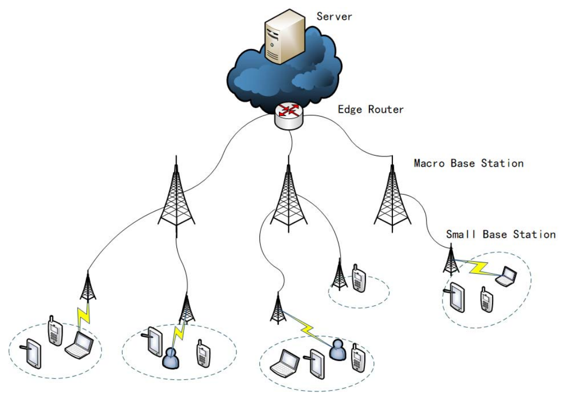

Figure 1. Architecture of 5G Networks

generates the more accurate result. Next, we transform data to be supervised learning and then transform on a specific scale. After the above work, we obtain the LSTM prediction model and the predicted outcomes based on this model. At last, we invert outcomes with scaling and differencing reversely.

IV. Caching Strategy

This section firstly introduces the architecture of the 5G network in our scenario. Base on the architecture, we present our caching strategy, which splits capacity into the proactive one and the reactive one, to improve hit ratio and reduce latency when requesting content. The proactive cache is used to prefetch popular content, while the reactive one caches others with the ordinary caching algorithm. It is obvious that proportion of the proactive cache affects the performance of our strategy. Therefore, we utilize simulated annealing algorithm to configure the proportion of the proactive cache in real time.

A. Architecture

As shown in Fig. 1, there are four levels in the network, edge router, macro base stations(mBSs), small base stations(sBSs) and user devices(UDs). Traditionally the mobile devices send requests to fetch content from the nearest sBS. If the requested piece of data has not been cached in the sBS, the request would be sent to the neighbor. If still not hit, sBSs send requests to the upper base station or servers where we assume all the content is stored.

B. Classification of Content

It is mentioned that the content is assumed to be limited within the items whose popularity is known priori because a fraction of content is frequently used. In practice, there must be new items whose popularity cannot be predicted because they have too little historical records to train prediction models. Accordingly, we classify all the content into two categories, old items and new items. As far as old items are concerned, we calculate the R^2 in order to evaluate the accuracy of prediction models. A data set has n values $y_i (i \in [1, n])$, each associated with a predicted value f_i. We always use the root mean squared error in formula 1 to measure the residual, but the magnitude of the observed value has a great influence on RSME.

$$RMSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - f_i)^2 \tag{1}$$

If $\bar{y}$ is the mean of the observed data, then

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - f_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

The closer to 1 the R^2 is, the more accurate the result is. We define a threshold to divide the prediction result of old items into two categories, accurate and inaccurate. The content whose R^2 of prediction is beyond the threshold is called predictable items, while new items and some old items whose predictions are inaccurate are regarded as unpredictable items.

C. Cache Partition Algorithm

The cache of each base station is split into the proactive cache and the reactive cache. The proportion of proactive cache is a decision which may have an influence on the efficiency of our caching strategy. After the prediction, the content is classified into two categories, predictable items and unpredictable items. At each period, we use the predicted popularity of predictable items and the requested count of unpredictable items at the last period as the predicted frequency to be requested. Each small base station has the table of the predicted frequency with the pair of item id and frequency.

As shown in Algorithm 2, we employ the idea of simulated annealing in adjusting the proportion of proactive cache. When the temperature is greater than the minimum temperature, we generate a new proportion which is a neighbor of the current proportion. For the current proportion and new proportion, we calculate the latency with the predicted frequency at the next period since latency reflects the quality of experience. If the latency of the new proportion is shorter than that of the current proportion, this new proportion would replace the current one. Otherwise, there would be a probability of adopting the new proportion. When completing annealing, the optimal proportion of the next period is produced. At the beginning of the next period, the proportion of proactive cache would be adjusted to the optimal one.

D. Heuristic Prefetching Caching Strategy

There have been various caching strategies for reducing repetition of transmissions. In our strategy, we additionally

Algorithm 2 Proportion Selection

```
Require:
  proportion at the current period, $X_{now}$;
  initial temperature,$T_0$;
  minimum temperature,$T_{min}$;
  decrease rate of temperature,$decr$;
Ensure:
  new proportion at the next period,$X_{new}$;
  $t = T_0$
  while $t > T_{min}$ do
    $Latency_{now} = getLatency(X_{now})$
    $X_{new} \Leftarrow neighborofX_{now}$
    $Latency_{new} = getLatency(X_{new})$
    $delta = Latency_{new} - Latency_{now}$
    if $delta < 0$ then
      $X_{now} \Leftarrow X_{new}$
    else
      $p = \frac{1}{10*(1+e^{delta/T})}$
      if $random() < p$ then
        $X_{now} \Leftarrow X_{new}$
      end if
    end if
    $t = t * decr$
  end while
```

consider about prefetching content which is high likely to be requested and placing these contents close to users. As a consequence, the transmission distance and repetition are reduced substantially, which leads to great improvements on both latency and hit ratio.

It is commonly understood that the more frequently a piece of content is requested, the more likely it will be requested. Thus we predict the popularity of content before the next period to prefetch popular items.

Furthermore, according to the Zipf's distribution, a few content items are requested frequently while the vast majority are requested rarely. As shown in the formula 2, the frequency P(r) has the opposite tendency for the rank of frequency r. It follows that prefetching a few popular content items in cache would gain excellent performance. However, the inaccurate prediction might cause bad selection of items to cache.

$$P(r) = \frac{C}{r^{\alpha}} \tag{2}$$

Each base station collects popularity of content and feeds it in prediction service. Small base stations would get the predicted frequency of each data item after the prediction. At the beginning of each period, the proportion of proactive cache would be adjusted to the optimal one with the cache partition algorithm. Within the limitation of the proactive cache capacity, each small base station needs to prefetch items for a sum total maximum of their frequency. In this step, we use the knapsack problem algorithm to obtain optimal selection. The Algorithm for sBSs is shown in Algorithm 3.

Algorithm 3 Popular Content Selection

```
$frequencyTable$ = Map of (id, frequency)
$SizeTable$ = Map of (id, size)
$selectedList$ = KnapsackSolve($frequencyTable$,$SizeTable$)
for each $content \in selectedList$ do
  proactiveCache.add($content$);
  frequencyTable.remove($content$);
end for
transfer $frequencyTable$ upward
```

At the beginning of each period, each small base station also prefetches the selected items its proactive cache from the server and transfers the remaining frequency table upward. The macro base station receives the remain of frequency table from every branch of small base stations and merges all the tables. Afterward, we also use the knapsack problem algorithm to select what to cache for the largest sum of frequency. There is no need to transfer the remain of frequency table upward since we assume that the server stores all the contents.

Once requests arrive at the sBSs, the proactive cache is firstly sought for the requested item. If the requested item has been cached in the proactive one, it can be sent back to the client directly. Otherwise, we need to look up on the reactive cache. If still fail to fetch the requested item, the request would be sent to the upper mSBs. Similarly, we would fetch from the proactive cache then turn to the reactive cache in mBSs. Provided that the lookup in the mBS is a miss, the request should be transferred to the server. In backhaul, the returned items would be placed into the reactive cache layer by layer.

V. EVALUATION

In this section, we first describe the scenario where we conduct the experiment and then list the parameters of the simulation. We present the accuracy of the prediction algorithm. Then we evaluate hit ratio and latency of our caching strategy and make a comparison with the archetype. The result demonstrates the fitness of heuristic prefetching caching strategy and reveals the circumstances where it is more worthy to applied.

A. Scenario

We evaluate heuristic prefetching caching strategy with the architecture of one server, three mBSs, and five sBSs, as the hierarchical relationship shown in Figure 1. With referring to EdgeCloudSim in [24], user devices send requests to sBSs through WLAN while the base stations communicate with each other with MAN connection. Besides mBSs connect to the server over the WAN.

Table I
SIMULATION PARAMETER

Parameter	Value
Simulation Time(hours)	36
Time period(hour)	1
Number of server/mBS/sBS	1/3/5
Number of content	40
The content size(M)	20
WAN/MAN/WLAN Bandwidth(Mbps)	20/100/300
mBS size/sBS size	3
sBS size	200, 400
Proportion of mBS	0.4

In our evaluation, all the contents have the same size. Despite that the content size is actually different, we assume that they could be divided into chunks of equal size. As for the improvement of performance, Least Recently Used(LRU) is not only the archetype for comparison but also the policy of reactive cache in our strategy as a representative sample of the ordinary caching algorithms. The parameter of simulation is listed in Table I.

B. Dataset

In order to measure the efficacy of our prediction algorithm and caching strategy, we applied our model to Requests dataset which is derived from YouTube videos dataset. YouTube videos dataset is an every hour real-time count observation(views, comments, likes, dislikes) during May 2018 of videos that are released in April 2018 on Youtube. At each hour, the number of views for each video is a good representative of the request count of each data item. However, the number of some videos is beyond 100 million, which causes a terrible waste of time to simulate a flood of requests in sequence. Therefore, we construct Requests dataset base on the trend of the views in YouTube videos dataset. For each data item, the number of requests at every hour is less than one thousand in Requests dataset.

C. Accuracy of Prediction

To evaluate the accuracy of prediction models, we calculate the R^2 of results. The closer to 1 R^2 is, the more accurate the predicted result is. In Figure 2, the predicted values approximate to the true values. Figure 3 presents the obvious differences of the predicted values and the true values. R^2 of the first prediction in Fig. 2 is about 0.999, while the second is 0.011 or so. We define a threshold to classify all the predictions. The prediction whose R^2 is over 0.9 is regarded as the accurate one, otherwise it is inaccurate.

D. Performance

To measure the efficiency of our caching strategy, we calculate hit ratio and latency and observe the proportion of each small base station. In my previous work, we assume that the percentage of predictable items collected by each

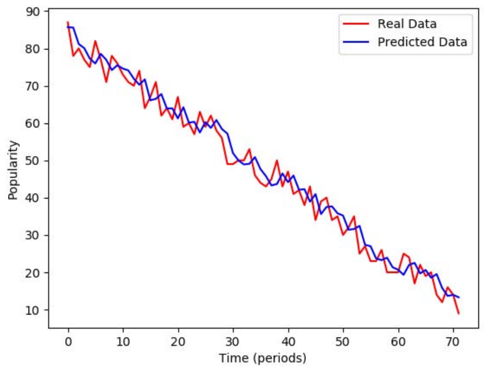

Figure 2. Accurate Predition

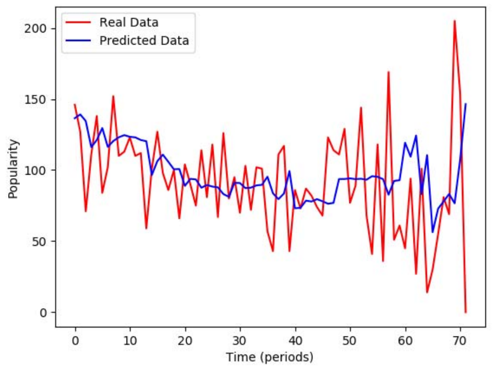

Figure 3. Inaccurate Prediction

small base station is the same. That is, the previous caching strategy is not applicable to the architecture which allows different percentages of predictable items at each base station. Besides, the proportion of each base station selected by caching strategy is also the same. However, heuristic prefetching caching strategy allows each small base station to adjust itself to the optimal proportion of cache partition at each period. Thus, the proportion of each small base strategy is always different. We first evaluate the latency using the same percentage of predictable items in each small base station. Then we demonstrate the adaptability of heuristic prefetching caching strategy with different percentages of predictable items.

We measure the hit ratio of heuristic prefetching caching strategy and the archetype with LRU caching algorithm. In Figure 4, the hit ratio of the LRU archetype and our caching strategy is similar when the percentage of predictable items is zero. With the percentage of predictable items increasing, hit ratio of LRU archetype decreases a little, while that of heuristic prefetching caching strategy increases greatly.

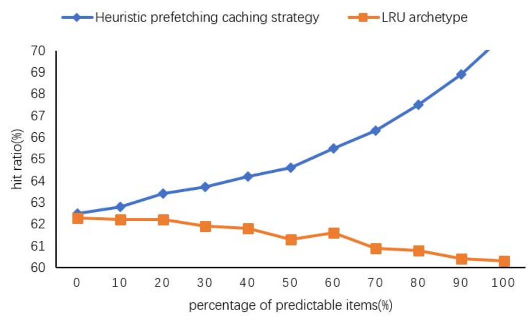

Figure 4. Hit ratio with 400MB sBS

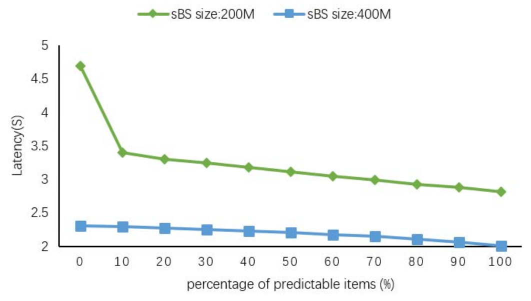

Figure 5. Latency with Heuristic Prefetching Caching Strategy

Providing that all the content is predictable, the improvement of hit ratio reaches 20% nearly.

When it comes to the quality of experience, it makes more sense to focus on latency. It is obvious in Figure 5 that latency would go down as the percentage of predictable items increases. To demonstrate the optimization, we calculate the improvement with the benchmark of traditional cache strategy which uses the LRU algorithm to cache.

As shown in Figure 5 and Figure 6 , the situation with more predictable items earns larger benefit from our caching strategy since the accurate prediction makes prefetching more effective. In other words, heuristic prefetching caching strategy is more suitable for circumstances that there are a great number of predictable items. Moreover, the sharp reduction of latency when the percentage of predictable items is from 0% to 10% illustrate that our caching strategy is almost useless when all items are unpredictable. To be precise, this strategy could improve latency by 52% in 200MB sBSs, 15% in 400MB sBSs if all the content is predictable. This result fills with common sense that worse situation has more room for improvement.

We try to discover the adjustment rule of proportion and find the tendency of proportion. However, it seems to be an endless process of mutual cooperation and continual readjustment. Table II shows a part of this process, which obviously holds no rules.

Comparing with the improvement of latency in my previ-

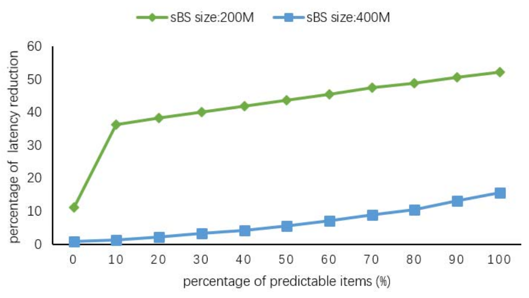

Figure 6. Percentage of Latency Reduction

Table II
PROPORTION OF EACH SBS

period	1	2	3	4	5	6	7	8
BS0	0.25	0.85	1	0.85	0.25	0.55	0.25	0.75
BS1	0.25	0.25	0.8	0.95	0.85	0.9	0.85	0.25
BS2	0.9	0.5	0.95	0.8	1.0	1.0	0.65	0.75
BS3	0.25	0.65	0.55	0.15	0.9	0.85	0.65	0.75
BS4	0.15	0.2	0.8	0.15	0.15	0.3	0.6	0.95

ous work, the performance of heuristic prefetching caching strategy is a little better but more practical because it is applicable to the situation that there are new items. Moreover, heuristic prefetching caching strategy also solve the problem that the previous strategy is not applicable to the situation that there are different percentages of predictable items at each base station.

We evaluate our caching strategy with the parameter that percentages of predictable items in five small base stations are 10%, 30%, 50%, 70%, 90%. Latency is reduced by around 8%, which is close to the enhancement when the percentage of predictable items is 70% at each sBS.

VI. CONCLUSION

It is generally believed that high quality of experience is always the urgent demand. However, the traditional reactive networking paradigm causes low hit ratio and high latency in meeting heavy traffic demands. In this paper, we propose heuristic prefetching caching strategy to improve quality of experience with the architecture of 5G networks. We split cache into the proactive one which prefetches popular content to maximize the sum of the frequency and the reactive one which caches others with the ordinary caching algorithm. We predict the popularity of content based on its historical frequency with LSTM time-series prediction algorithm in order to provide the predicted frequency for the proactive cache to prefetch popular content.

To demonstrate the efficiency of this strategy, we evaluate hit ratio and latency every ten percent of predictable items in the architecture of 200MB, 400MB sBSs. Heuristic prefetching caching strategy gains the improvement of hit ratio by 20% and the reduction of latency by 52% in 200MB sBS.

Moreover, we draw the conclusion that this strategy is more suitable for the larger percentage of predictable items.

An extension of our caching strategy would be adjusting the proportion of the macro base stations to the optimal one by themselves at each period. Besides, we would apply the idea of simulated annealing to obtain the optimal proportion by the historical latency of every period rather than the calculated latency with predicted frequency at one period.

Acknowledgment

This paper is supported by Project 213.

References

[1] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5g wireless networks," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 82–89, 2014.

[2] F. Zhang, C. Xu, Y. Zhang, K. Ramakrishnan, S. Mukherjee, R. Yates, and T. Nguyen, "Edgebuffer: Caching and prefetching content at the edge in the mobilityfirst future internet architecture," in *World of wireless, mobile and multimedia networks (WoWMoM), 2015 IEEE 16th International Symposium on a*. IEEE, 2015, pp. 1–9.

[3] A. Sengupta, S. Amuru, R. Tandon, R. M. Buehrer, and T. C. Clancy, "Learning distributed caching strategies in small cell networks," in *Wireless Communications Systems (ISWCS), 2014 11th International Symposium on*. IEEE, 2014, pp. 917–921.

[4] D. Liu, B. Chen, C. Yang, and A. F. Molisch, "Caching at the wireless edge: design aspects, challenges, and future directions," *IEEE Communications Magazine*, vol. 54, no. 9, pp. 22–28, 2016.

[5] C. Bernardini, T. Silverston, and O. Festor, "Mpc: Popularity-based caching strategy for content centric networks," in *Communications (ICC), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3619–3623.

[6] E. Bastug, J.-L. Guénégo, and M. Debbah, "Proactive small cell networks," in *Telecommunications (ICT), 2013 20th International Conference on*. IEEE, 2013, pp. 1–5.

[7] J. Tadrous, A. Eryilmaz, and H. El Gamal, "Proactive content download and user demand shaping for data networks," *IEEE/ACM Transactions on Networking*, vol. 23, no. 6, pp. 1917–1930, 2015.

[8] A. Mahmood, C. Casetti, C.-F. Chiasserini, P. Giaccone, and J. Harri, "Mobility-aware edge caching for connected cars," in *Wireless On-demand Network Systems and Services (WONS), 2016 12th Annual Conference on*. IEEE, 2016, pp. 1–8.

[9] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási, "Limits of predictability in human mobility," *Science*, vol. 327, no. 5968, pp. 1018–1021, 2010.

[10] F. A. Gers and E. Schmidhuber, "Lstm recurrent networks learn simple context-free and context-sensitive languages," *IEEE Transactions on Neural Networks*, vol. 12, no. 6, pp. 1333–1340, 2001.

[11] O. Salman, I. Elhajj, A. Kayssi, and A. Chehab, "Edge computing enabling the internet of things," in *Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on*. IEEE, 2015, pp. 603–608.

[12] R. K. Sitaraman, M. Kasbekar, W. Lichtenstein, and M. Jain, "Overlay networks: An akamai perspective," *Advanced Content Delivery, Streaming, and Cloud Services*, vol. 51, no. 4, pp. 305–328, 2014.

[13] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. Leung, "Cache in the air: exploiting content caching and delivery techniques for 5g systems," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 131–139, 2014.

[14] B. Blaszczyszyn and A. Giovanidis, "Optimal geographic caching in cellular networks," in *Communications (ICC), 2015 IEEE International Conference on*. IEEE, 2015, pp. 3358–3363.

[15] N. Golrezaei, A. F. Molisch, A. G. Dimakis, and G. Caire, "Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution," *IEEE Communications Magazine*, vol. 51, no. 4, pp. 142–149, 2013.

[16] G. Gürsun, M. Crovella, and I. Matta, "Describing and forecasting video access patterns," in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 16–20.

[17] B. Bharath, K. Nagananda, and H. V. Poor, "A learning-based approach to caching in heterogenous small cell networks," *IEEE Transactions on Communications*, vol. 64, no. 4, pp. 1674–1686, 2016.

[18] D. Niu, Z. Liu, B. Li, and S. Zhao, "Demand forecast and performance prediction in peer-assisted on-demand streaming systems," in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 421–425.

[19] Z. Wang, L. Sun, C. Wu, and S. Yang, "Guiding internet-scale video service deployment using microblog-based prediction," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 2901–2905.

[20] M. Rowe, "Forecasting audience increase on youtube," 2011.

[21] S. Li, J. Xu, M. Van Der Schaar, and W. Li, "Popularity-driven content caching," in *Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on*. IEEE, 2016, pp. 1–9.

[22] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Fifteenth annual conference of the international speech communication association*, 2014.

[23] Y. Cui, S. Ahmad, and J. Hawkins, "Continuous online sequence learning with an unsupervised neural network model," *Neural computation*, vol. 28, no. 11, pp. 2474–2504, 2016.

[24] C. Sonmez, A. Ozgovde, and C. Ersoy, "Edgecloudsim: An environment for performance evaluation of edge computing systems," in *Fog and Mobile Edge Computing (FMEC), 2017 Second International Conference on*. IEEE, 2017, pp. 39–44.