Volume: 4, Issue: 4
Page: 104-112
2020

ISSN 2520-4750 (Online) & ISSN 2521-3040 (Print)

International Journal of Science and Business

Content Caching Strategy at Small Base Station in 5G Networks with Mobile Edge Computing

Rahman Atiqur, Ali Md Liton, & Guangfu Wu

Abstract:
The way of using mobile community sources successfully by the content material caching strategy. Here, we have a look at Small Cell Base Station Networks (SCBSNs) from a caching perspective. Firstly, we wager that these Small Cell Base Stations are deploying with low doable backhaul links on the other hand have excessive functionality storage units. Then, we illustrate our system mannequin and define a Quality of Experience (QoE) metric in order to satisfy a given file request. After that, we formulate the optimization hassle in order to maximize the Quality of Experience (QoE) metric for all requests beneath the ability constraints. We make clear this trouble via introducing an algorithm, referred to as MyCaching, which relies on potential constraints. Due to storage constraints, not all requested files can be cached so MyChacing algorithm selects the most popular documents until the total storage potential is achieved. In a row, the proposed caching algorithm is compared with PropCaching, the numerical consequences illustrate that the range of content material requests increases. Besides, we exhibit that MyCaching performs higher than PropCaching in most cases. Such as, for R=200 wide variety of requests and a storage ratio γ=0.5, the pleasure in MyCaching Algorithm is 90% higher than PropCaching and the backhaul usage is condensed by using 5%.

IJSB
Accepted 08 April 2020
Published 09 April 2020
DOI: 10.5281/zenodo.3746233

Keywords: Small Cell Base Station Networks (SCBSNs), proactive caching, popularity caching, Quality of Experience (QoE), Mobile Edge Computing (MEC).

About Author (s)

Rahman Atiqur, (Corresponding Author), Ph.D., Information and Communication Engineering, Chongqing University of Posts and Telecommunications, Chongqing, 400065, P.R. China.

Ali Md Liton, Ph.D., Information and Communication Engineering, Chongqing University of Posts and Telecommunications, Chongqing, 400065, P.R. China.

Guangfu Wu, Senior Engineer, School of Software Engineering, Chongqing University of Posts and Telecommunications, Chongqing, 400065, P.R. China.

1 Introduction

Todays, market forecasts point out an explosion of wireless site visitors. The traffic generated by wired units is a good deal lesser than that of the visitors generated by way of wireless devices. Even though, the long term evolution (LTE) as brand new superior networks, mobile networks will now not be capable to sustain the demanded rates. To overcome this lacking, small cell phone base station networks (SCBSNs) have been proposed as a answer, and they are predicted to be the successor of Long Term Evolution (LTE) networks. Deploying such excessive statistics rate SCBSNs to fulfill this demand wishes high-speed devoted backhaul. For costly nature of this requirement, the present day nation of the artwork proposes to add excessive storage gadgets (i.e., hard-disks, solid-state drives) to small cells base station and use these units for caching functions. For decreasing the price and additive benefits, the work in proposes to make use of such a dense infrastructure opportunistically for cloud storage eventualities both for caching or continual storage. In parallel, every other line of research focuses on content caching to deal with network assets efficaciously and can be seen as a complementary. Here, we complementarily merge the two methods for SCBSNs. We focal point on SCBSNs the place we have small base stations deployed with high storage gadgets however have confined backhaul links. In fact, we have the following annotations:

In common networks, which are referred to as here reactive networks, user requests are relaxed right after they are initiated. Compare with proactive SCBSNs can track, learn and then create a person request prediction model. So, we get flexibility in scheduling efficient resources. Although human things to do is incredibly predictable and correlated, actually, predicting the correct time of person requests may no longer be achievable. Nevertheless, statistical patterns such as file reputation distributions can also help to allow a positive stage of prediction. Doing this, the estimated documents can be cached in SCBSNs. So, the backhaul can be offloaded and movable customers can have a higher stage of satisfaction. The caching in SCBSNs can be cheap like the storage devices have become incredibly cheap. Such as, putting two terabytes of storage in a SCBS charges approximately one hundred twenty dollars. Generally the network operators install obvious caching proxies to speed up carrier requests and limit bandwidth expenses. We take a look at that these types of caching procedures can be a complementary way of value discount by means of migrating the position of these proxies to SCBSNs. Given these annotations, the intention of this work is to examine the affect of caching in SCBSNs. Searching for caching as a way of being proactive; we also introduce an algorithm known as MyCaching. The approach relies on the recognition data of the requested files. We then examine our outcomes with PropCaching.

Road Map: We discuss the problem scenario and describe our System model in Section 2. We formulate an optimization hassle and clarify the MyCaching algorithm in Section 3 Related numerical outcomes are given in Section 4 Finally, we conclude in Section 5.

The mathematical notation used in this paper is the following. Lower case and uppercase italic symbols (e.g., a, A) are scalar values. A lower case boldface symbol (e.g., a) denotes a row vector and an upper case boldface symbol (e.g., **A**) a matrix. $\mathbf{1}_{M\times N}$ represents an $M \times N$ matrix with all ones, and $\mathbf{0}_{M\times N}$ is again $M\ x\ N$ sized matrix but with entries set to zeros. $\mathbf{\Lambda}$ (a) is a diagonal matrix constructed from vector **a**. The indicator function 1{.>.} returns 1 when the inequality holds, otherwise 0. Finally, the transpose of **a** is denoted with $\mathbf{a}^t$.

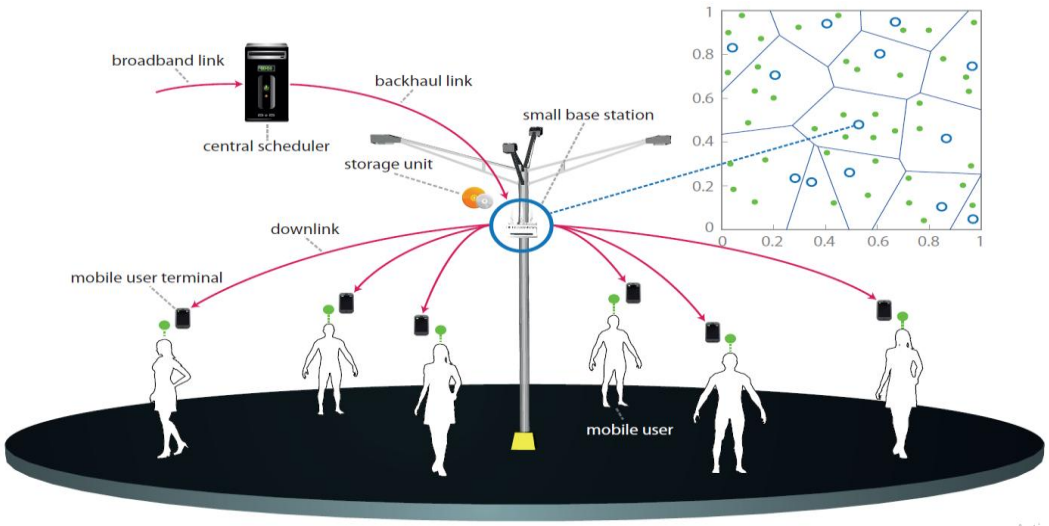

Figure 1. A SCBSNs state of affairs consists of a central scheduler, small cells and user terminals

2 System model

Consider a SCBSNs scenario where a central scheduler (CS) and M SCs are in charge of serving N user terminals (UTs) as depicted in Figure 1. In this setting, the CS is coordinating and providing broadband access to SCs over cheap backhaul links with given capacities $\mathbf{a} = [a_1, ..., a_M] \in \{0, Z^+\}$. On the other hand, SCs have high storage units with the storage capacities $\mathbf{s} = [s_1, ..., s_M] \in \{0, Z^+\}$, thus, they can cache information coming from the CS and serve their UTs over wireless links with the rate of:

$$R = \begin{bmatrix} R1 \\ R2 \\ . \\ . \\ Rm \end{bmatrix} = \begin{bmatrix} R1,1 & \cdots & R1,n \\ \vdots & \ddots & \vdots \\ Rm,1 & \cdots & Rm,n \end{bmatrix} \in \{0, Z^+\}^{MxN} \quad (1)$$

where $R_{i,j}$ represents the rate from i-th SC to j-th UT, in bits per timeslot.

Now, suppose that over a given time window T, users want to perform requests with the rates specified by $\mathbf{q} = [q_1, ..., q_N] \in \{0, Z^+\}$. In other words, the j-th user wants to perform q_j number of requests during the time window T.

Let us say for instance that users want to download files from internet. Thus, the CS keeps track of F different files indexed as $\mathbf{f} = [f_1, ..., f_F]$. Each file f_i is atomic and has a length specified by l_i. We denote $\mathbf{l} = [l_1, ..., l_F] \in Z^+$.

In a reactive scenario, requests would be satisfied by the CS just after they are initiated by the user. In a proactive scenario, the CS tracks, learns and then predicts the user requests before the actual request arrival. This helps to decide which files in peak time where the load of the backhaul is very high. Thus it would avoid large delay in file delivery. Let us assume discrete popularity distributions of files of UTs:

$$P = \begin{bmatrix} \bar{p}_1 \\ \bar{p}_2 \\ . \\ . \\ \bar{p}_N \end{bmatrix} = \begin{bmatrix} \bar{p}_{1,1} & \cdots & \bar{p}_{1,F} \\ \bar{p}_{2,1} & & \bar{p}_{2,F} \\ \vdots & \ddots & \vdots \\ \bar{p}_{N,1} & \cdots & \bar{p}_{N,F} \end{bmatrix} \in \{0,1\}^{NxF} \quad (2)$$

where $\bar{p}_{i,j}$ represents the probability of the j-th file being requested by the i-th UT. Note that the matrix $\bar{P}$ is row-stochastic as i-th row represents the discrete probability distribution of i-th user, hence, the sum of all elements in i-th row equals to 1. The $\bar{P}$ might be sparse as number of files grows by time.

As we defined above, $\bar{P}$ represents the local file popularities of the UTs. The file popularity distributions observed at the SCs will be different than $\bar{P}$as all connected users will contribute with their local file popularities. To show this, let us first define the connectivity matrix, such that,

$$C=\begin{bmatrix} C_1 \\ C_2 \\ \cdot \\ \cdot \\ C_N \end{bmatrix} = \begin{bmatrix} C_{1,1} & \cdots & C_{1,N} \\ C_{2,1} & & C_{2,N} \\ \vdots & \ddots & \vdots \\ C_{M,1} & \cdots & C_{M,N} \end{bmatrix} \in \{0,1\}^{MxN} \quad (3)$$

where $c_{m,n} = 1\{w_{m,n} > 0\}$. Then, we can derive popularity distributions at the SCs, called P, by multiplying $\bar{P}$ with the user request rates q, the connectivity matrix C, and a normalization factor. The equation is given in (4), where $p_{i,j}$ represents the popularity of the j-th file seen by the i-th SC. If the matrix P can be obtained by CS at t = 0, proactive caching strategies can be employed. In practice, this matrix can be computed at the CS by counting the number of times the files are requested. This would give the information about the file popularity of the future requests as the user behavior is correlated. Assuming that the matrix P is perfectly given in our scenario, the following step is to decide how to distribute these files among SCs. This is detailed in the next section.

Finally, we assume that the content popularity follows a uniform distribution where each content is equally popular. As a result, each SBS randomly selects and caches contents from the library.

3 Proactivity through caching

In this section, we formulate an optimization problem for the satisfied requests over total requests and provide a MyCaching algorithm to heuristically maximize the objective. Suppose that, over the time window T as assumed in the scenario, the CS is observing a number of R file request events listed in $r = [r_1; :::; r_R]$. A request $r_i \in \mathbf{r}$ is done by a UT to download the file fri with the corresponding length of l^{ri} . Hence, the CS has to deliver this file to the user either from the internet or from the caches of the connected small cells. The delivery for the request r_i starts at $t = t^{ri}$ and finishes until the file is completely downloaded by the UT at $t=\hat{t}^{r_i}$. For each file f_i, let us define its corresponding bandwidth requirement

$$\begin{aligned} \mathbf{P} &= \mathbf{\Lambda}\left(\frac{1}{\mathbf{c}_1\mathbf{q}^T}, \ldots, \frac{1}{\mathbf{c}_M\mathbf{q}^T}\right) \mathbf{C}\mathbf{\Lambda}(\mathbf{q})\bar{\mathbf{P}} \\ &= \underbrace{\begin{bmatrix} \frac{1}{\mathbf{c}_1\mathbf{q}^T} & 0 & \cdots & 0 \\ 0 & \frac{1}{\mathbf{c}_2\mathbf{q}^T} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \frac{1}{\mathbf{c}_M\mathbf{q}^T} \end{bmatrix}}_{\text{normalization factor}} \underbrace{\begin{bmatrix} c_{1,1} & \cdots & c_{1,N} \\ c_{2,1} & \cdots & c_{2,N} \\ \vdots & \vdots & \vdots \\ c_{M,1} & \cdots & c_{M,N} \end{bmatrix}}_{\text{connectivity}} \underbrace{\begin{bmatrix} q_1 & 0 & \cdots & 0 \\ 0 & q_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & q_N \end{bmatrix}}_{\text{request rates}} \underbrace{\begin{bmatrix} \bar{p}_{1,1} & \cdots & \bar{p}_{1,F} \\ \bar{p}_{2,1} & \cdots & \bar{p}_{2,F} \\ \vdots & \vdots & \vdots \\ \bar{p}_{N,1} & \cdots & \bar{p}_{N,F} \end{bmatrix}}_{\text{popularity distribution of files at UTs}} \quad (4) \\ &= \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \vdots \\ \mathbf{p}_M \end{bmatrix} = \begin{bmatrix} p_{1,1} & \cdots & p_{1,F} \\ p_{2,1} & \cdots & p_{2,F} \\ \vdots & \vdots & \vdots \\ p_{M,1} & \cdots & p_{M,F} \end{bmatrix} \in [0,1]^{M \times F}. \end{aligned}$$

Algorithm MyCaching for File Replacement

Input: s,f,l //This is the input of this algorithm

1: O: = 0_{MxF}, s:= 0_{Mx1} // Initializes Cache decision matrix O and the current storage vector s.

2: for a = 1....M do

[i,j] := (f_a) // Returns i and j as order values and indices.

3: for b=1.....F do

k:= j_b // Gets index of b-th most popular file.

if $l_k + s_a \leq s_a$ then // Checks if the b-th most popular file can be stored in the a-th SC.

$O_{a,k}$;= 1 // Set the element of the cache decision matrix to 1, meaning that the file will be cached.

$S_a := s_a + l_b$ // Increase the current storage.

else

break

4: end if

5: end for

6: end for

Output: O

as $d_i \in \mathbf{d} = [d_1; :::; d_F]$. Now, suppose that $d^{r_i} \in \mathbf{d}$ is the bandwidth requirement for the request r_i and $x^{r_i}(t)$ is the amount of delivered information up to the time t (see Figure 2). By definition we consider that the request r_i is satisfied, if the average delivery rate in any time instance is superior to d^{r_i}. In this setting, the following formula holds:

$$\frac{1}{\hat{t}^{r_i} - t^{r_i}} \sum_{t=t^{r_i}}^{\hat{t}^{r_i}} \mathbb{1}\left\{\frac{x^{r_i}(t)}{t - t^{r_i}} \geq d^{r_i}\right\} = 1. \qquad (5)$$

The idea in (5) is to ensure that the network delivers the data with enough speed such that waiting is not going to occur during the playback. Therefore, if the request is satisfied, UT will have a better quality of experience (QoE). The determination of the value d^{r_i} might be different for different types of content. Roughly speaking, watching a compressed high definition (HD) video on a website (i.e., YouTube) requires 3-4 Mbps of bandwidth for interruption-free playback. For our scenario, an optimization problem can be formulated in the sense of maximizing the number of satisfied requests under the capacity constraints. If we denote $\hat{R}$ as the ratio of the satisfied requests over the total requests R, or simply the satisfaction ratio.

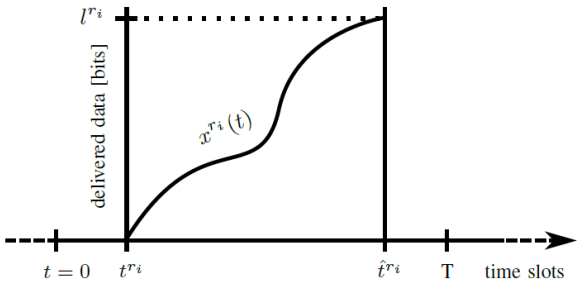

Figure 2: Deadline of a request r_i

We have,

$$\begin{aligned} \underset{t^{r_i}}{\text{maximize}} \quad & \hat{R} = \frac{1}{R} \sum_{r_i \in \mathbf{r}} \frac{1}{\hat{t}^{r_i} - t^{r_i}} \sum_{t=t^{r_i}}^{\hat{t}^{r_i}} \mathbb{1} \left\{ \frac{x^{r_i}(t)}{t - t^{r_i}} \geq d^{r_i} \right\} \\ \text{subject to} \quad & \mathbf{b} \preceq B^{max}, \\ & \mathbf{s} \preceq S^{max}, \\ & \mathbf{W} \preceq W^{max}, \end{aligned} \tag{6}$$

Where B^{max}, S^{max} and W^{max} are the capacity constraints of backhaul links, storage units and wireless links, respectively.

In general, even being able to predict the request times and storing the corresponding files in SCs before their arrival, all requests might not be satisfied due to the capacity constraints. Employing a brute-force search by varying the start of the delivery times would be hard due to the combinatorial behavior of the problem. Therefore, instead of a time oriented approach, we focus on caching the popular files to achieve a certain amount of proactivity. Suppose that the CS has a cache decision matrix (also called caching matrix), such that

$$\mathbf{O} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ . \\ . \\ \theta_M \end{bmatrix} = \begin{bmatrix} \theta_{1,1} & \cdots & \theta_{1,F} \\ \vdots & \ddots & \vdots \\ \theta_{M,1} & \cdots & \theta_{M,F} \end{bmatrix} \in \{0,1\}^{MxF} \tag{7}$$

where $\theta_{i,j} = 1$ means that the i-th SC has to store the j-th file, and $\theta_{i,j} = 0$ is the vice versa. The complete proactive case in our scenario would be achieved if all the files could be cached in the SCs. Hence, our caching matrix would be $\mathbf{O} = \mathbf{1}_{M \times F}$, meaning that all SCs cache all files before the requests start. In the worst case, no file would be stored, so the caching matrix would be $\mathbf{O} = \mathbf{0}_{M \times F}$.

The matrix $\mathbf{O}$ in CS can be obtained using the MyCaching algorithm, as given in Algorithm 1. The algorithm basically chooses to store the files with the highest popularities until the storage capacity of SCs are achieved. The detailed explanation is given step by step in the algorithm. Recall that this cache method of decision is due to the fact that the exact request times are not practically obtainable. Even knowing the request times and then solving the problem would be hard with a brute-force search algorithm. Our heuristic approach is used to maximize our objective by caching the files according to the popularity statistics of the files and the storage capacities of SCs.

Assuming that the complexity of the initialization step is $0(1)$ and the sorting operation has $0(F \log F)$ worst-case complexity (i.e., by using Timsort), the complexity of Algorithm 1 becomes $0((1 + M(F \log F + 4F)) \simeq (M F \log F)$, which is linear in M and almost linear in F. As we present MyCaching in a simple way for sake of clarity, in real-time systems where M

and F is very big and the time window is moving, a more efficient method could be implemented by employing an iterative approach of the algorithm.

4 Numerical results

A discrete event simulator was implemented in order to obtain the numerical results for the scenario. The capacities of the backhaul links are assumed to be lower than the capacity of the wireless links. Over the time window T, the request times are pseudo-randomly generated with uniform distribution. The popularity of files is also generated using uniform distribution. The simulation parameters are given in Table I. After generating the request times with respect to the parameters, Algorithm 1 is simulated for different values of the storage ratio, which is defined as

Table I: SIMULATION PARAMETERS

Parameter	Values	Description
T	2024	time window (time slot)
M	8	number of SCs
N	32	number of UTs
B^{max}	32	total capacity of backhaul links (Mbit/time slot)
W^{max}	256	total capacity of wireless links (Mbit/time slot)
F	256	number of files
$l_i, \forall i$	512	length of files (Mbit)
$b_i, \forall i$	8	backhaul link capacities (Mbit)
$w_{i,j}, \forall i,j$	4	wireless link capacities (Mbit)
$d_i, \forall i$	8	QoE requirement (Mbit)

$$\gamma = \frac{S^{max}}{M \sum_{i=1}^{F} l_i} \qquad (8)$$

In order to see the impact of the storage in SC with the proactive caching approach, the following γ values are simulated: {0; 0:25; 0:50; 0:75; 1}. In the worst case, when $\gamma = 0$, SCBSNs would have zero storage capacity. Hence, Algorithm 1 would compute the caching matrix as $O = 0_{MxF}$. On the other hand, $\gamma = 1$ is the best case where each SCBSNs has enough storage to cache all requested files. Therefore the caching matrix would be $O = 1_{MxF}$. After estimating the caching matrix for each γ values using Algorithm 1, the corresponding files are assumed to be in SCBSNs in order to avoid additional bandwidth usage for their delivery to SCBSNs. In fact, this assumption is reasonable as some files might be either previously stored or can be stored on their first delivery. After this assumption, by iterating over time, the CS delivers the files either from the internet or from the cache of the connected SCs. In case of simultaneous transmissions due to the same request times and/or continuing transmissions, the network resources are fairly shared among the UTs. More precisely, the total backhaul and wireless bandwidths are equally divided among the requests. The simulation using MyCaching is repeated while increasing the number of requests. These operations are repeated 100 times and averaged. To compare the gain of MyCaching, simulations of the PropCaching method is also performed by filling the O with uniform distribution, until the storage capacity is achieved.

Numerical results for MyCaching are shown in Figure 3(a). The figure shows the evolution of the satisfaction ratio $\hat{R}$ over the total requests R. For each γ value, the satisfaction ratio remains 1 until a certain threshold because of sufficient capacity of the links. After that, due to the congestion caused by many simultaneous deliveries, the number of satisfied requests

starts to decrease and converges to 0. This is obvious due to the fair bandwidth sharing policy. As the total requests R → ∞, the amount of delivery bandwidth given to a request tends to 0. Therefore, no request will be satisfied as R → ∞. The decrement in case of γ = 0 is strictly related to the backhaul capacity as the files are delivered over the backhaul. The more we increase γ the more we store the files in SCs. Thus, the decrease becomes more dependent on the wireless link capacities. In general, as γ increases, the satisfaction ratio $\hat{R}$ becomes better compared to the non-caching case. Figure 3(b) illustrates the backhaul bandwidth consumption over the total requests R. It is seen that as increases, the amount of bandwidth consumption decreases. This is evident as more files are cached.

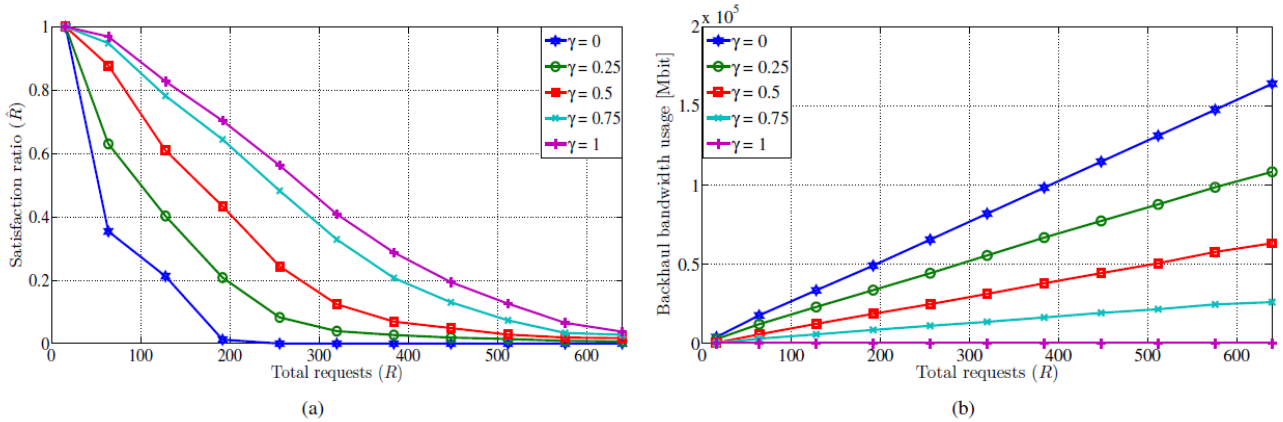

Figure 3: Numerical Results of MyCahing: (a) evolution of the satisfaction ratio, (b) the backhaul bandwidth usage

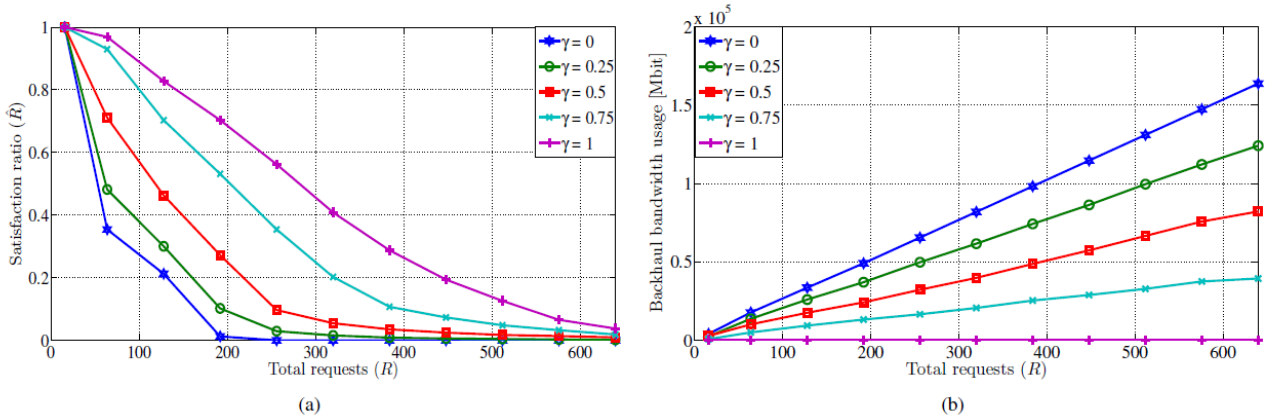

Figure 4: Numerical Results of PropCaching: (a) evolution of the satisfaction ratio, (b) the backhaul bandwidth usage

For the performance of PropCaching, Figure 4(a) and Figure 4(b) depict the evolution of the satisfaction ratio and the backhaul bandwidth consumption respectively. In case of γ = 0 and γ= 1, the performance is obviously identical to MyCaching. However, in between, MyCaching has different gains for different R values. For example, for R = 200 and γ = 0.5, the satisfaction of MyCaching is 90% higher than PropCaching and the backhaul usage is reduced by 5%.

5 Conclusions

Through this article, proactive SCBSNs equipped with low capacity backhaul links and high storage units are investigated from a caching perspective. By using the popularity statistics of the files and employing a caching strategy based on this, the impact of storage in SCBSNs is studied. As the load of the network increases by the number of requests, our results show that caching has a better performance in satisfying the requests compared to the non-caching case. Moreover, MyCaching outperforms PropCaching in the most cases. Other approaches of proactivity could be caching the files according to the recent and trending statistics of the files. We think that by storing recently downloaded files or trending files according to time varying behavior of the file popularities, the performance of these networks can be improved.

6 References

E. Bas¸tug˘, J.-L. Guénégo, and M. Debbah, (2012) "Cloud storage for small cell networks," in IEEE CloudNet'12, (Paris, France).

H. E. Gamal, J. Tadrous, and A. Eryilmaz,(2010) "Proactive resource allocation: Turning predictable behavior into spectral gain," in Communication, Control, and Computing (Allerton), (2010) 48th Annual Allerton Conference on, pp. 427 – 434.

J. Tadrous, A. Eryilmaz, and H. E. Gamal,(2011) "Proactive resource allocation: Harnessing the diversity and multicast gains," submitted to IEEE Transactions on Information Theory, arXiv preprint: 1110.4703.

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

V. Etter, M. Kafsi, and E. Kazemi,(2012) "Been There, Done That: What Your Mobility Traces Reveal about Your Behavior," in Mobile Data Challenge by Nokia Workshop, in conjunction with Int. Conf. on Pervasive Computing.

Fayek, Jana, et al.(2018) "Device-to-Device Communication in 5G: Towards Efficient Scheduling." International Journal of Digital Information and Wireless Communications 8.3 (2018): 144-150.

Cite this article:

Rahman Atiqur, Ali Md Liton, & Guangfu Wu (2020). Content Caching Strategy at Small Base Station in 5G Networks with Mobile Edge Computing. *International Journal of Science and Business, 4*(4), 104-112. doi: https://doi.org/10.5281/zenodo.3746233

Retrieved from http://ijsab.com/wp-content/uploads/517.pdf

Published by

IJSAB International