International Journal of Computer Engineering & Technology (IJCET)
Volume 9, Issue 5, September-October2018, pp. 190–200, Article ID: IJCET_09_05_022
Available online at
http://www.iaeme.com/IJCET/issues.asp?JType=IJCET&VType=9&IType=5
Journal Impact Factor (2016): 9.3590(Calculated by GISI) www.jifactor.com
ISSN Print: 0976-6367 and ISSN Online: 0976–6375

MOBILE CACHE SHARING AND PRE-FETCHING FOR LATENCY REDUCTION USING SIGNALR

P. Amudha Bhomini
Research Scholar St. Xavier's College, Palayamkottain Manonmaniam Sundaranar University Abishekapatti,

Dr. Jayasudha J.S
Department of Computer Science and Engineering Sree Chitra Thirunal College of Engineering, Trivandrum

ABSTRACT

The people expect contents to be available anytime, anywhere instantly using smart phones. Unlike client server environment, mobile devices have limited resources like cache memory, network bandwidth, power usage etc, when user expects a superior experience where contents are available instantly, there is a need for lot of researches in providing contents quickly using caching techniques, pre-fetching contents and also share cache at various levels like server level, network level, nodes, among collaborative users between clients (other smart phones/ mobile devices) etc. There are various researches going on to improve cache hits and try to provide web contents using only cache and perfected data, without hitting the server too many times for contents. In this paper we propose a combination of caching and pre-fetching using Enhanced Bloom Filter by individual mobile devices to improve hit ratio as well as share this cached content with set of users who voluntarily register to the hub (group). This method was implemented using SignalR and provided the evaluation. Using this method users themselves can establish voluntary groups (called cooperative users) and share their cached contents among cooperative users. The sharing of cached content from the group/hub can improve their cache hit rates for delivering content locally, reduce latency and bandwidth requirements.

Keywords: mobile caching, latency reduction, pre-fetching, client cache sharing, cooperative, collaborative cache sharing, SignalR.

Cite this Article: Dr. Jayasudha J.S, P. Amudha Bhomini. Mobile Cache Sharing and Pre-Fetching for Latency Reduction Using Signalr *International Journal of Computer Engineering & Technology,* 9(5), 2018, pp.190–200.

http://www.iaeme.com/IJCET/issues.asp?JType=IJCET&VType=9&IType=5

1. INTRODUCTION

Caching and pre-fetching techniques have been used since the advent of Internet. There are many techniques used in client server environment, distributed network environment and many techniques were introduced and improved with advent of improvements in data mining, neural networks, cloud computing, nodes and network sharing, cluster based sharing, collaborative sharing etc.

However, with smart phones being available with everyone, with the same challenge of making contents available quickly to the mobile users with limited resources and with mobility has also attracted lot of researches and novel implementations.

There are many research works on caching and pre-fetching to improve hit rate in caching like (COCA), pre-fetching [1], Adaptive pull and push algorithm for content and cluster based data consistency approaches [2], Few cluster based cooperative or non cooperative or adaptive schemes for improving caching efficiency [3], Performance Improvement of Cache Management [4] etc.

There are many studies and implementations of cache for mobile device using bloom filters which is a very effective structure for implementing mobile cache. Many researches are done to improve the efficiency for data, distributed images [5], auto scaling of data [6], extracting sample set from Bloom Filter and reconstruction [7] as well as self adjusting bloom filter construction [8]. There are many bloom filter variations on applications of Bloom Filter in the field of network security [9].

Various studies are done on pre-fetching techniques to improve hit rate and reduce latency using predictive pre-fetching [13], Objective Optimal Algorithms[10], improve pre-fetching performance[11], client centric approaches for latency minimization using pre-fetching [12] as well as many evaluations done on web pre-fetching algorithms [14].

Few researches are done on the combination of caching and pre-fetching [15], cluster based pre-fetching [16], combination of caching, pre-fetching and collaborative caching and pre-fetching [17]. This paper contains a hybrid method of Enhanced Bloom Filter (EBF) for caching and pre-fetching and uses SignalR for sharing cached data with cooperative users who registers with the hub. It reduces web traffic, bandwidth consumption and latency, and increases the cache hit ratio.

The rest of the paper is organized as follows: Section 2 deals with enhanced bloom filter technique for caching and pre-fetching, Section 3 describes data sharing among cooperative users, Section 4 describes the implementation using SignalR and Section 5 presents the results and discuss the performance evaluation and Section 6 concludes the paper of the proposed solution.

2. ENHANCED BLOOM FILTER TECHNIQUE FOR IMPLEMENTING EFFECTIVE CACHE AND PRE-FETCHING

Enhanced Bloom Filter methodology (EBF) for caching use Message Digest (MD5) and bloom filter (BF) techniques, replacing the web pages using Least Recently Used (LRU) integrated with First in First Out (FIFO) algorithm and pre-fetching web pages based on user log and bandwidth. These caching and pre-fetching techniques are highly recommended to increase cache hit ratio and to reduce latency for accessing the web pages almost instantly.

The different processes involved in caching/pre-fetching using enhanced bloom filter technique are given below

1) Accept user request
2) Apply MD5 algorithm for each URL
3) Convert the hex value into its equivalent binary value

4) Maintaining the bloom filter array
5) Calculating the cache hit ratio and maintaining cache
6) Pre-fetch the anticipated web pages based on user log and available bandwidth and store it in pre-fetch area
7) Adopt LRU integrated with FIFO as the cache replacement policy

The flow chart for enhanced bloom filter technique used for web catching/pre-fetching is given in figure 1.

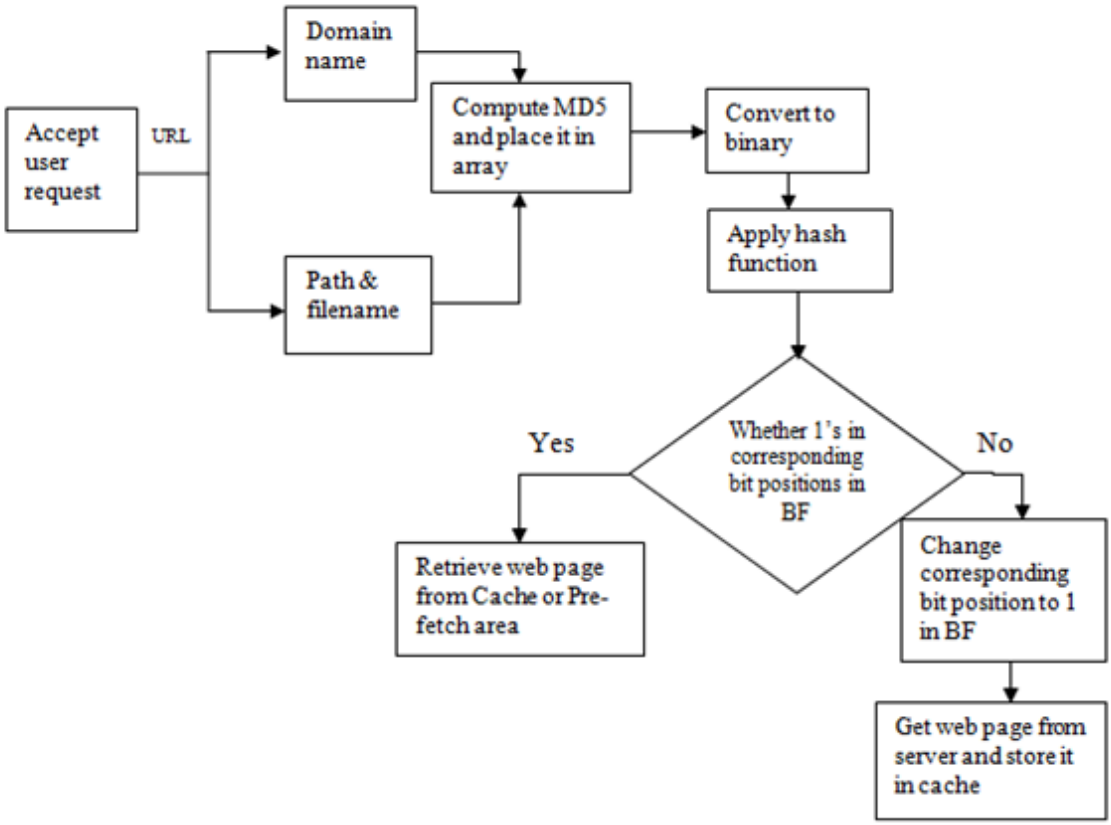

Figure 1 Flow chart of Enhanced Bloom Filter Technique

3. CACHE DATA SHARING AMONG COOPERATIVE USERS

Data in cache and pre-fetching area using Enhanced Bloom Filter can be used to render content from individual devices. This improves the performance much better in terms of latency, available network bandwidth etc. However in the above method if content not found in cache / pre-fetch area as explained in the previous section, content need to be obtained from server and then stored in cache using LRU with FIFO so that it can be rendered further.

This can be further improved, if there is a set of cooperative users are established and content is checked with their cache / pre-fetch area (using same enhanced bloom filter) rather than checking with the server. The cooperative cache data sharing is used along with Enhanced Bloom Filter for caching and pre-fetching. This has considerable improvement in cache hit ratio and if the content is available with any one cooperative user, then server hits are totally avoided.

The context diagram for establishing hub connection with Cooperative Users (CUs) is given in figure 2.

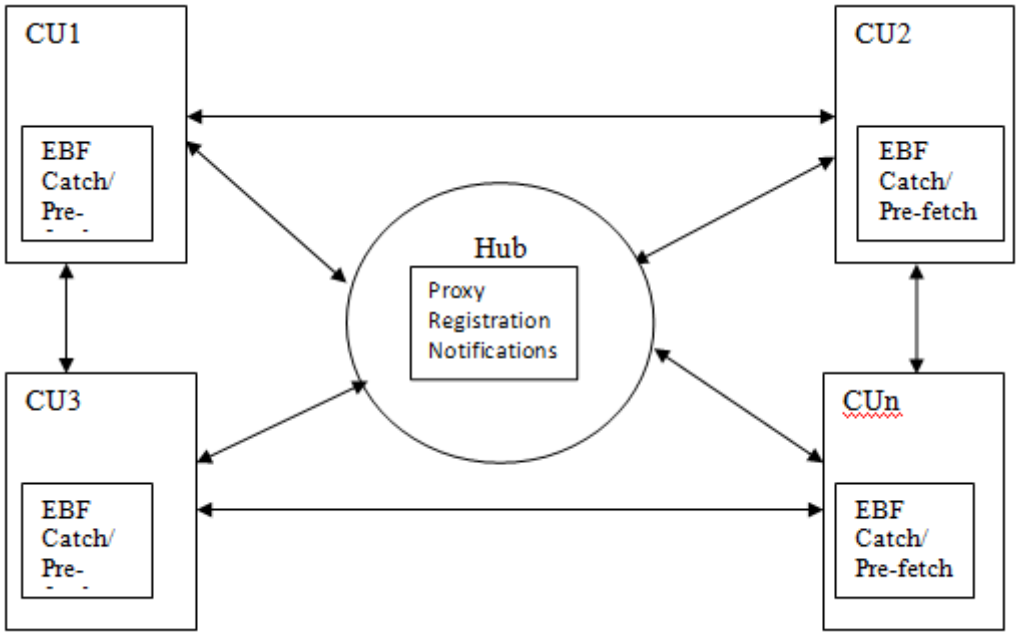

Figure 2. Context diagram for Hub and CUs

The proposed methodology is given below

There is a central (URL) application (hub) to establish proxy for individual users / devices to join the cooperative users.

Prerequisites

1. Users install the application to implement EBF and pre-fetching as well as to join the hub to collaborate
2. Keep the application enabled to collaborate.
3. Whenever hit the server to render data, notify all collaborative users.

Continuous Process

4. When a URL is requested, search in cache and pre-fetch area
5. If hit render it locally, update hit count
6. If miss, increase miss count in local cache and search notifications from cooperative users
7. A) If notifications received from cooperative users (check the notifications) then request the specific collaborative user for content.
    a. Get data rendered by the specific cooperative user
    b. Update cache (EBF)
    c. Update weightage
    d. Update hit count in hub

B) If notification not received from any CUs

    e. Forward the request to the server and render data
    f. Publish notification
    g. Update cache (EBF)
    h. Update weightage

In this approach, caching and pre-fetching techniques are integrated along with the contents from cooperative users. If cooperative users have similar browsing behavior then over a period most of the data can be rendered locally. In this method, chances of rendering data without accessing from the server is very high.

The flow chart for data sharing among cooperative users is given in figure 3.

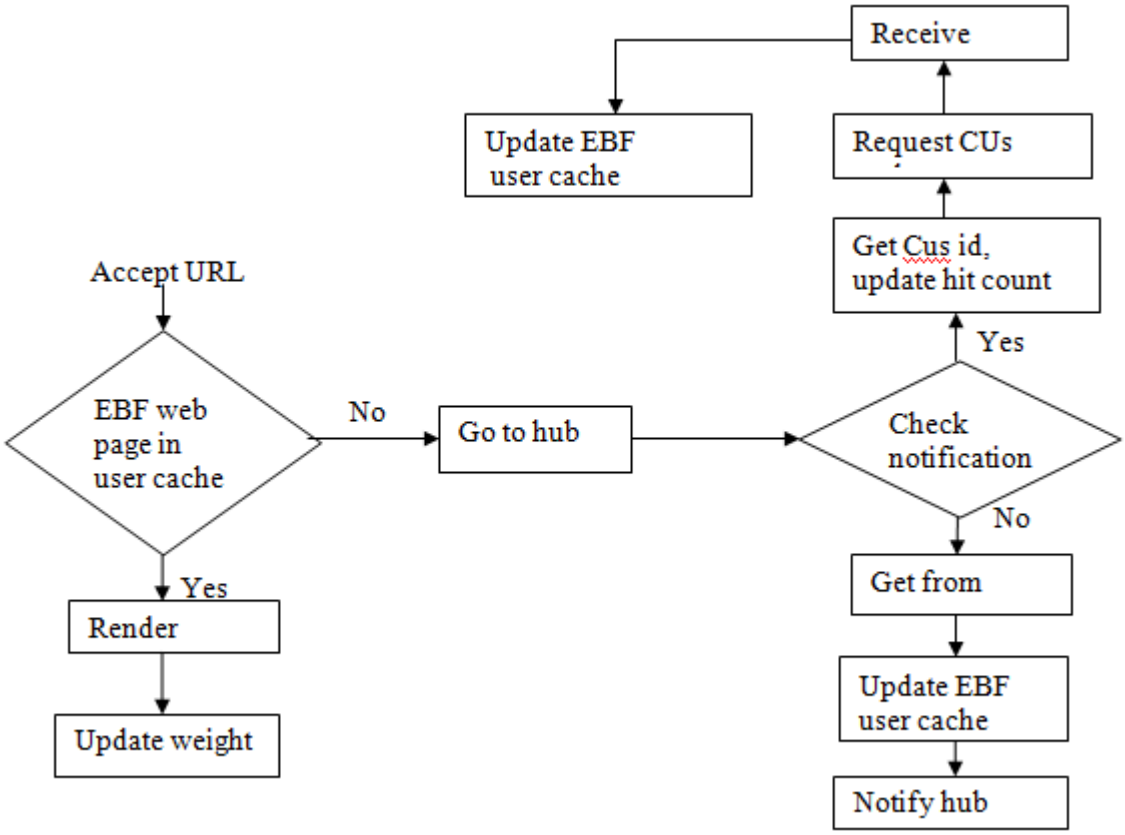

Figure 3 Data sharing among cooperative users

4. IMPLEMENTATION OF CACHE SHARING AMONG COOPERATIVE USERS USING SIGNALR

SignalR is a new ASP.NET library, which uses existing transport technologies based on the infrastructure. SignalR has the capability of real-time communication with wide range of clients [18]. SignalR is a library for ASP.NET to add real-time web functionality to applications. Real-time web functionality has the ability to push content to the connected clients as it happens, in real-time.

Sharing data among co-operative users

The different phases involved in the process of sharing data cached by EBF among cooperative users are given below.

i. Creation of Hub connection
ii. Creation of hub proxy
iii. Start Connection
iv. Group join
v. Notification
vi. Send/Receive data
vii. Update EBF Array

i. Creation of hub connection

The signalR hub is used to make Remote Procedure Calls (RPCs) from a server to a connected client and from a client to the sever.

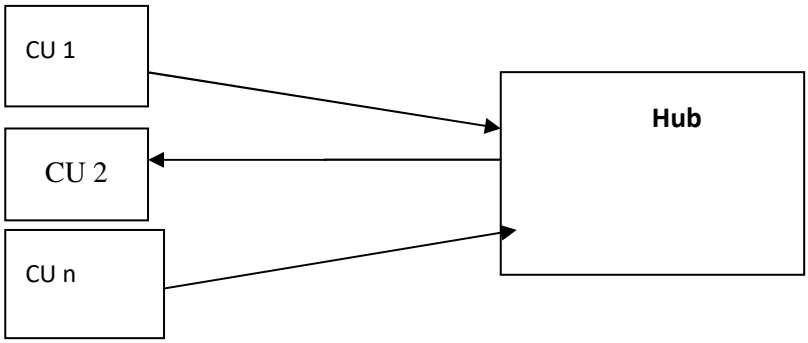

Figure 4 Establishing hub connection

Hub connection is initialized using a Uniform Resource Locator (URL). This URL helps all the users to get connected with hub.

ii. Creation of hub proxy

Once the hub connection is established, hub proxy is created with a name. All data will be transmitted through proxy.

iii. Connection start

After initializing the hub and hub proxy, connection can be made between hub and mobile devices (clients/cooperative users). This helps to send/receive data among users.

iv. Group join

The CUs who want to join in the group can join the group, then register their details. The registration has to be done using device ID and name of the user. All CUs information are stored in the hub. If anyone wants to get the information, the information can be obtained from the hub.

Once CUs joined in the hub, they can start communicating with the hub when they receive new data in their cache or when they need data which is not in their cache. This is done via notifications and via send/receive data.

v. Notification

Notification can be given to a specific user or group (all users). It can be given by domain name or URL. Notifications show that the available/requested information.

vi. Send/Receive data

The data can be transmitted to a single user or group after getting the notifications. Once the request is received, first it is checked whether the domain name/URL is available in CUs Enhanced Bloom Filter (EBF) array. If the domain name is available in EBF all relevant cached/pre-fetched data correspond to the domain will be used locally, if not it is requested to the hub and then to the respective CUs based on the notifications received in the hub. If a specific URL is available in CUs EBF, only the data related to that URL alone will be sent to the requested user. If domain is requested, all relevant data will be sent.

Techniques used to send data are given below.

1) Serialization using JSON (Java Script Object Notification) format

2) Encryption/Decryption using Advanced Encryption Standard (AES)
3) Compression

i) JSON format

It is a minimal readable format for structuring data. It is used to transmit data between server and client. It takes less space as compared to eXtensible Markup Language (XML) or any other format.

After converting the data into its corresponding JSON format the total size of the web page has to be calculated. If the size of web page is more than available buffer size then page has to be split according to the buffer size and it is sent with indexing.

ii) Encryption

For encryption, Advanced Encryption Standard (AES) algorithm is used, which is one of the most common and widely used symmetric block cipher algorithm for providing security. It is extremely difficult for hackers to get the real data when encrypting by AES algorithm [19]. It has the ability to deal with three different key sizes 128, 192 & 256 bits. In the proposed techniques, AES is used to encrypt the data using 128 bit key.

iii) Compression

Gzipstream (compress) method in .Net is used to compress the data. Thus it takes less bandwidth, while transmission and reduces the web traffic/latency.

The techniques used to receive the data at the receiver side are given below.

1) Deserialization using JSON
2) Decryption
3) Decompression

i) Deserialization using JSON

At the receiver side, first it has to check whether indexing is available. If it is available, then the number of pages have to be retrieved and all the serialized data have deserialized. If not it deserializes the data once.

ii) Decryption

It is the process of getting the original data from cipher text. The sender and receiver uses the same key to encrypt and decrypt data.

iii) Decompression

The data is decompressed at the receiver side using signalR.

vii) Update EBF Array

The received data is stored in receiver's cache and the EBF array is updated with this new URL, so that from next time onwards the subsequent request for the same URL can be served from local cache without accessing the data from web server or from the cache of other cooperative users.

The above process is continued in all cooperative users for all URLs.

5. PERFORMANCE EVALUATION

The performance of the proposed technique is evaluated and analyzed in terms of latency.

Latency is the time delay between the request and response. The following figures show the time delay when accessing the web pages corresponding to the URL's between single user and co-operative users.

Figure 5 shows the comparison of latency between EBF and without caching techniques.

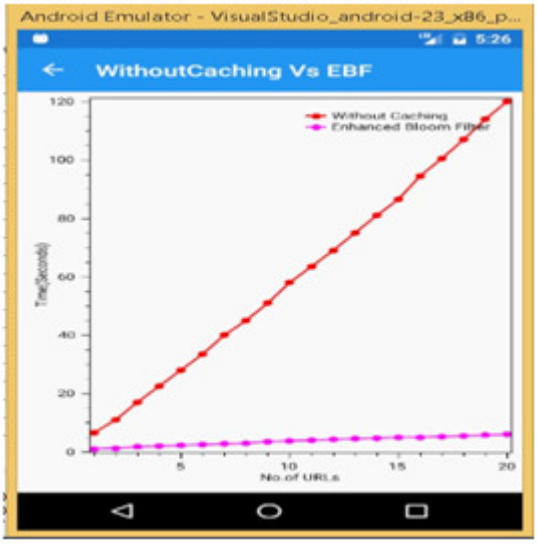

Figure 5 Analysis of latency between EBF and without caching technique

It has been observed that latency is more when no EBF is implemented and latency is reduced considerably using EBF technique.

The figure 6 shows the comparison of latency between the different techniques such as no EBF or Cooperative caching, Caching and pre-fetching along with Cooperative caching.

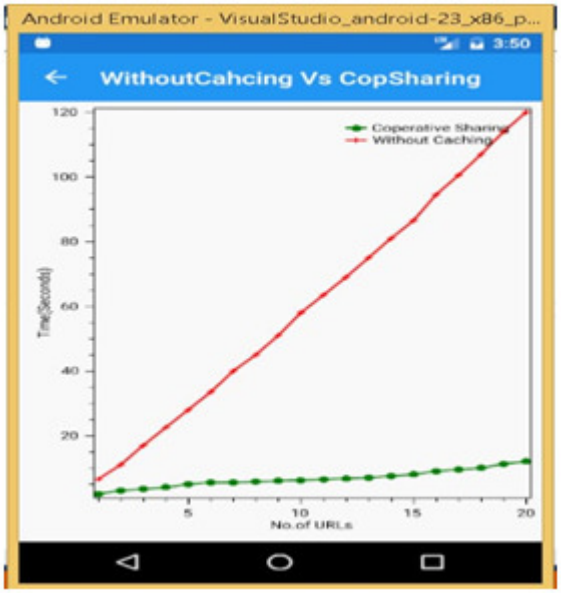

Figure 6 Analysis of latency between without caching and cooperative caching techniques

Figure 7 shows the comparison of latency between EBF and cooperative caching techniques.

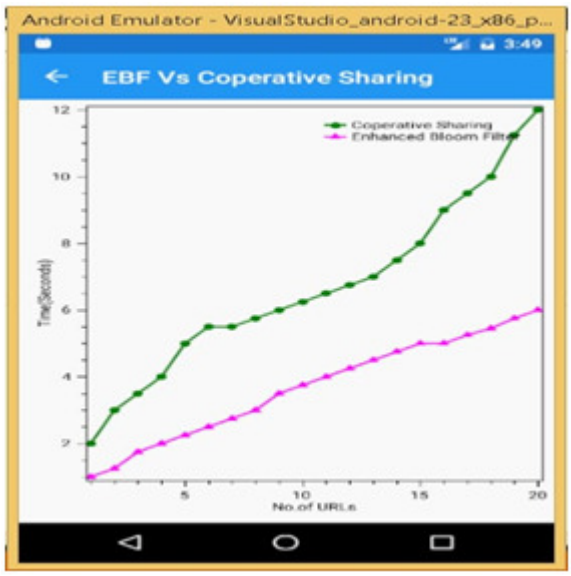

Figure 7 EBF Vs Cooperative caching

It has been observed that latency is more when no EBF is incorporated and with cooperative sharing, latency was reduced considerably, even more than only with EBF. In order to confirm the performance, all 3 different scenarios are considered in figure 8

The analysis of latency has also been done using the following techniques.

a. Without caching
b. With only EBF
c. With EBF and Cooperative caching

The figure 8 shows that the latency is very less if rendered from local EBF, little more if rendered from cooperating CUs EBF and high if it is from server as seen in next graph.

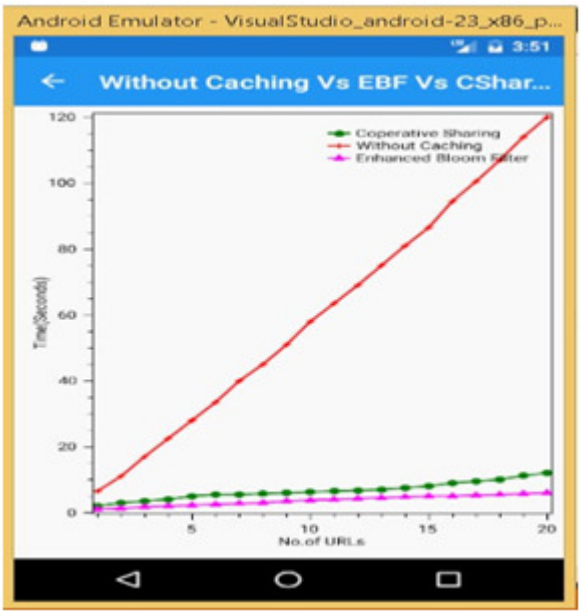

Figure 8 Analysis of latency between without caching and EBF and cooperative caching techniques

6. CONCLUSION

The Caching and pre-fetching techniques using Enhanced Bloom Filter (EBF) gives better experience to user perceived latency. In addition, creating a hub of cooperative users to share catched data further reduces latency, if data are not available in its own cache, but available with other cooperative users. In this case, the latency is much better than rendering data from server. In this proposed technique, the cache sharing is optimized by registering cooperative users and making participation voluntary. Enhanced BF for cache management as well as cache sharing between cooperative users, without EBF has been implemented. A well defined process has been used for fetching data from local EBF or from cooperative users. Necessary statistics like hit and miss rate in local EBF as well as cooperative users cache are recorded for further analysis and optimization.

REFERENCES

[1] Shashidhara, D.N., Chandrappa, D.N., "A Literature Survey on Caching and Pre-fetching Techniques in MANET", Journal of Information Technology & Software Engineering, 6(5), 2016.

[2] Kuppusamy, P. and Kalaavathi, B., "Cluster Based Data Consistency for Cooperative Caching over Partitionable Mobile Adhoc Network", American Journal of Applied Sciences, Science Publications, 9(8), 2012, pp. 1307 – 1315.

[3] R.Sarada, Shirin Banu Koduri, Dr. M.Seetha, "A Cluster Based Cooperative Caching Mechanism in MANET", International Journal of Computer Science and Technology, 3(2), 2012, pp. 1057-1062.

[4] Abdulaziz Zam, Movahedinia, N., "Performance Improvement of Cache Management in Cluster Based MANET", I.J. Computer Network and Information Security, 10, 2013, pp. 24-29.

[5] Andrea Salvi, Simone Ercoli, Marco Bertini and Alberto Del Bimbo, "Bloom Filters and Compact Hash Codes for Efficient and Distributed Image Retrieval", IEEE International Symposium on Multimedia, 2016, pp. 515 – 520.

[6] Dennis Kleyko, Abbas Rahimi, Ross W. Gayler and Evgeny Osipov, "Auto Scaling Bloom Filter Controlling Trade-off Between True and False Positives", arXiv:1705.03934v2, 2017.

[7] Neha Sengupta, Amitabha Bagchi, Srikanta Bedatthur, Maya Ramanath, "Sampling and Reconstruction using Bloom Filters", IEEE International Conference on Data Engineering, 2017, pp. 195-198.

[8] Gil Einziger and Roy Friedman, "Tiny Set – An Access Efficient Self Adjusting Bloom Filter Construction", IEEE/ACM Transactions on Networking, 2017.

[9] Shahabeddin Geravand and Mahmood Ahmadi, "Bloom Filter Applications in Network Security a State of the art Survey", Elsevier, Computer Networks, 2013, pp. 4047-4064.

[10] Bin Wu and Ajay, D., Kshemkalyani, "Objective Optimal Algorithms for Long-Term Web Pre-fetching", IEEE Transactions on Computers, 55(1), 2006.

[11] Josep Domenech, Jose A. Gil, Julio Sahuquillo, Ana Pont, "Using current web page structure to improve perfecting performance", Elsevier, Computer Networks, 2010, pp. 1404 – 1417.

[12] Yixue Zhao, "Toward Client Centric Approaches for Latency Minimization in Mobile Applications", IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems, 2017, pp. 203 – 204.

[13] Venkata N. Padmanabhan, Jeffrey C. Mogul, "Using Predictive Pre-fetching to Improve World Wide Latency", ACM SIGCOMM, Computer Communication Review, pp. 22-36.

[14] Joseph Domenech, Ana Pont, Julio Sahuquillo, Jose A. Gil, "A user focussed evaluation of web pre-fetching algorithms", Elsevier, Computer Communications, 2007, pp. 2213-2224.

[15] Bathsheba Parimala, A., Jefferson, B., Appasamy, K., "Data Caching and Pre-fetching using Coollaborative Caching and Pre-fetching Algorithms in Wireless ADHOC Networks", International Journal of Engineering and Techniques, 3(6), 2017, pp. 35-41.

[16] Niklesh Rathore, Ms. Swati Tiwari, "New Framework for Web Proxy Server through Cluster Based Prefetcing", International Journal of Software & Hardware Research in Engineering, 2(10), 2014, pp. 1-4.

[17] Nanhay Singh, Arvind Panwar and Ram Shringar Raw, "Enhancing the Performance of Web Proxy Server through Cluster Based Pre-fetching Techniques", IEEE, 2013, pp. 1158 –1165.

[18] Anurag Choudhry, Anshu Premchand, "Real Time Apps Using SignalR", International Journal of Computer Trends and Technology, 15, 2014, pp. 92-96.

[19] Ako Muhamad Abdullah, "Advanced Encryption Standard (AES) Algorithm to Encrypt and Decrypt Data", Cryptography and Network Security, 2017.