I.J. Information Engineering and Electronic Business, 2017, 1, 36-46
Published Online January 2017 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijieeb.2017.01.05

HitBand: A Prefetching Model to Increase Hit Rate and Reduce Bandwidth Consumption

Islam Anik, Akter Arifa and Hamid Md. Abdul
American International University of Bangladesh, Dhaka, Bangladesh
Email: {abhi.cse.aiub, arifa.anisha123}@gmail.com, hamid@aiub.edu

***Abstract*—**Caching is a very important issue in distributed web system in order to reduce access latency and server load. A request is a hit if it is available in the cache and if not then it will fetch from the server in order to cache and serve the request. Researches have shown that generic algorithms of caching can increase hit rate up to 40−50%, but adding prefetching scheme can increase this rate to 20%. Prefetching is a technique to fetch documents before they are requested. This paper proposes a process model for prefetching named HitBand which will balance hit rate bandwidth in every scenario with the combination of "Roulette-wheel selection". Roulette-wheel selection is a very popular selection based algorithm which selects objects according to their fitness. We have compared our HitBand with the generic algorithms of prefetching like prefetching by popularity, apl characteristic, good Fetch and lifetime. Generic algorithms did not take web object size into consideration and in limited bandwidth scenario object size has a big impact on bandwidth consumption. Though prefetching by lifetime algorithm shows little concern about bandwidth consumption by getting the object with changes happening less frequently but this compromises the hit rate. But our proposed HitBand not only considers bandwidth but also hit rate during prefetching. Performance evaluation of HitBand along with other algorithms is provided in our paper. We have tested our HitBand with the testing engine which is built using JavaScript and maintained under AngularJS framework. From the performance evaluation, our HitBand shows better results both in high and low bandwidth.

***Index Terms*—**Prefetching, caching, roulette-wheel selection, distributed system, hit rate, bandwidth, distributed web based system.

I. Introduction

The World Wide Web (WWW) is an information space in which documents, images and other resources are identified by uniform resource locators (URL), interlinked by hypertext links and can be accessed via internet [1]. The usage of www is cheap and accessing information is faster than using any other means. The www has documents that solicit to a wide range of interest, for example news, education, scientific research, sports, entertainment, stock market growth, travel, shopping, weather and maps [2]. Both internet and www have experienced remarkable growth in past decade. With the passing of time, www is getting popular. Along with the increase of popularity of www, traffic over internet has also increased. According to the World Wide Web Size (www.worldwidewebsize.com), the indexed web contains at least 4.65 billion pages and the dutch indexed web contains at least 249.61 million pages. As WWW continues its march with the exponential growth, the major issues faced by the current web users are the network congestion and server overloading. Though capacity of the internet increases 45−60% every year, it is not enough to serve the demand for bandwidth as more and more services and utilities are moved into the web [3]. Also the uses of smartphones and tablet is increasing day by day. Within 2017, 55% of traffic will come from these devices [4]. Currently researcher has proposed a luxurious idea named internet of things (IoT) in which every device will get an IP address and they will be connected with each other via internet [5]. IoT provides power to the connected devices to think, see, hear, communicate, make decisions, execute jobs by talking together etc. [6]. Likely IoT, more and more schema will be added to internet in foreseeable future. If some kind of steps are not taken soon to solve the problems caused from the increase of the usage of the www, the www will become too congested and it will lose its attraction.

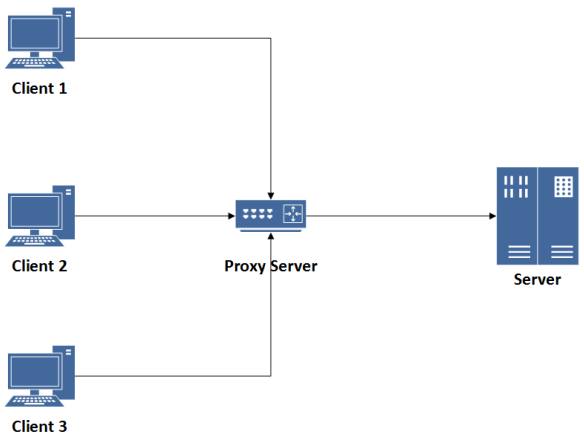

Fig.1. Client server communication via proxy server.

Many researchers have been working on improvement of the performance of Web since 90s. From their research, many approaches have been proposed [7, 37]. Among the proposed approaches, the web caching technology is proved as one of the effective solutions towards reducing

access latency (the perceived amount of time between a user sends a request and receives with a response), alleviating web service bottlenecks, decreasing server loads and increasing improvement of the scalability and quality of service of the web systems [8-11].

Web caching is a technology for the temporary storage of web objects (document, html pages, images, audio files etc.) for future retrieval. The major advantages of Web caching are

- It reduces bandwidth consumption as data is available locally, so requests do not have to go to the server.
- It reduces the server load as few requests will be passed to the server and most of the request will be handled locally from the cache.
- It reduces access latency as data will be always closed to the client, so that it can serve immediately.

Web proxies has gained rapid growth in recent years [10]. Web object can be cached at different locations between the client and the server [12-13]. From the location of caching, web caching can be divided into three parts

- **Browser caching** is built in by default in modern browsers. Browser utilizes clients RAM, CPU, Local disc to manage caches. Data of caching system cannot be share with other users as caching is done locally.
- **Proxy caching** is located between client and server [14]. Web proxy server can share data with multiple clients. Fig. 1 provides a structure of communication from client to server via proxy server. From [15], when a request comes to a proxy server, it first checks it's availability in the cache. If no objects is found, it will pass the request to the web server. After getting response from web server, it first caches the fresh copy and pass the data to the client. Basically proxy caches is located close to the clients. The purpose of proxy caching is to reduce access latency.
- **Server caching** is also located between clients and server but apart from proxy cache, this caching is located close to the server. The purpose of server cache is to alleviate server's workload.

Although web caching improved the performance of web but benefits from this technique is limited [16]. Previous researches have shown that maximum caching hit rate by applying any caching algorithm can achieve is less than 40−50% [3, 17-20]. In practical, one out of two objects cannot be found in caching [17]. To further increase of cache hit rate is to do prefetching the web objects into the cache but prefetching can increase the traffic [21]. Prefetching is a technique where documents are fetched or downloaded from server before they are requested. It improves user experience by loading a webpage faster as it already available in the cache with the help of prefetching [22]. Many researches have described that caching mixed with prefetching can double its performance rather than caching without prefetching [23-24]. According to [3], combination of perfect caching and perfect prefetching can reduce 60% client latency. Web prefetching has two main components like prediction engine and prefetching engine [25]. Prediction engine chooses objects according to the objects basic information like access frequency, lifetime etc. and prefetching engine takes the decision to prefetch them or not. Prefetching can be applied in three ways

- Between the browser and the web proxy.
- Between the web proxy and the web server.
- Between the client and the web servers.

Prefetching techniques can be divided into two types [26]

- **Short-time prefetching** (In this technique, it fetches objects which can be requested in near future based on user recent activity.)
- **Long-time prefetching (**In this technique, it fetches objects based on steady state object update frequency, access rate, lifetime etc.)

A proper prefetching depends on a good prediction based algorithm to select web objects. A precise selection can reduce access latency, on the contrary inaccurate fetching would lead to waste of bandwidth. Holland [27] developed an algorithm named "Roulette Wheel Selection (RWS)" a selection based algorithm which selects individuals according to their fitness. Better the fitness, the less chances to be not selected among the set of objects. This is similar to a Roulette wheel in a casino. Usually each portion of the wheel is assigned to each of the possible selected objects based on their fitness. It can be achieved by diving the freshness factor of a selected object, by the total freshness factor of the all the objects and normalize them to 1. After that selection is made based on how the wheel is rotated. The higher fitness of the object will less chance to be eliminated. From this process some weakest objects may survive at the end of the process but these objects could prove usefulness following the recombination process [28]. We mapped our proposed equation with the selection probability of RWS to generate opportunity for selecting an object based on their fitness.

The remaining of the paper is organized as follows. Section II describes an overview of the characteristics of web objects along with the review of other popular prefetching algorithms like prefetch by Popularity [29], Good Fetch [30], APL characteristic [31], and Lifetime [31]. Section III provides the analysis of the steady state hit rate and bandwidth consumption. Section IV describes performance evaluation model based on H/B model. Section V presents our proposed protocol along with the description of RWS. Section VI presents performance evolution and results. Finally, Section VII concludes with the summery of our work.

II. Overview

In this section, we discuss the characteristics of web objects along with existing prefetching algorithms and the description of the notations.

Table 1. Notations and their description

Notation	Description
S	Set of web objects
a	The total access rate
k	Bandwidth constant
$ff(i)$	Fitness function of i^{th} object
p_i	Access frequency of i^{th} object
l_i	Lifetime of i^{th} object
s_i	Size of i^{th} object
$f(i)$	Freshness factor of i^{th} object
h_i	Hit rate of i^{th} object
b_i	Bandwidth of i^{th} object
Hit_{pref}	Hit rate with prefetching
Hit_{demand}	Hit rate without prefetching
BW_{pref}	Bandwidth with prefetching
BW_{demand}	Bandwidth without prefetching

A. Notations and their description

Description of all the notations are provided in the Table 1.

B. Characteristic of web object

Prefetching process requires object's basic information like size, access frequency and lifetime to decide which object to pick. Researchers have found that access pattern of web pages follows Zip's distribution [32]. According to zip's law, relative probability of i^{th} popular object to get requested is inversely proportional to *i*. If access frequency of i^{th} most popular object is p_i, zip's law can be expressed as

$$k = \frac{z}{i} \text{ where } z = \frac{1}{\sum_i i}$$

Cunha et al. [33] have researched that among **N** web pages, the relative probability of i^{th} most popular object get requested is

$$p_i = \frac{z}{i^{\alpha}} \text{ where } z = \frac{1}{\sum_{j=0}^{N} (\frac{1}{j^{\alpha}})} \quad (1)$$

The value of α varies in different research. Cunha et al. [33] suggested a value of 0.986 and Nishikawa et al. [34] suggested 0.75 which is based on their 2000000 requests access log. Objects lifetime is also another characteristics that affects hit rate and bandwidth consumption. An object lifetime is denoted as the average time interval between the consecutive updates of the web object. Crovella et al. [35] described that Static web object's sizes follow a Pareto distribution characterized by a heavy tail. Crovella and Barford [36] have shown that the dynamic object's sizes follow a mixed distribution of heavy-tailed Pareto and lognormal distribution.

C. Existing Algorithms

1) Prefetch by Good Fetch

Venkataramani et al. [30] approached a criterion named Good fetch which balance the web object's update frequency and access frequency. According to this algorithm, the web objects whose poses the highest probability of being accessed during their average lifetime are the worthy candidate for prefetching. Suppose for the object *i*, if access frequency of that object is p_i, overall access rate *a* and the average lifetime l_i, the probability of being prefetched during its life is

$$p_{goodFetch} = 1-(1-p_i)^{al_i} \quad (2)$$

According to this algorithm, it prefetches a collection of web object whose $p_{goodFetch}$ value exceeds a certain threshold value which provides a natural way to keep down the bandwidth wastage by prefetching. The motive behind this algorithm is that the objects with comparatively longer update interval and higher access frequency are nominated to be prefetched and this criterion supposes to balance the bandwidth and hit rate and with that intention it increases the hit rate with a tolerable increase in bandwidth.

2) Prefetch by Popularity

In this algorithm, the process of prefetching, select *k* most popular objects to prefetch and maintain a copy of them in the cache. If a new object joins to the set of the most popular object or any of the old objects get updated, the system pulls the new objects into the cache. Markatos et al. [29] proposed a "Top Ten" approach for prefetching objects. According to this criterion, each server keeps the records of all accessed object and among of them, top ten popular objects are pushed into the cached whenever they are updated. Thus, this way server always keeps the top ten most popular object fresh.

3) Prefetch by APL characteristic

Jiang et al. [31] suggested another approach for prefetching the web object. Suppose for object *i*, if access frequency of that object p_i, overall access rate *a* and the average lifetime l_i then they proposed $ap_i l_i$ as the measurement of prefetching web objects. Those objects will be prefetched whose apl value exceeds a certain threshold value. Object's apl value represents the number of possibility of this object to being accessed during its average lifetime. The higher apl value of objects, the higher chance to being accessed during its lifetime. Thus, increase the chance of improving overall hit rate by prefetching these web objects.

4) Prefetch by Lifetime

In this algorithm, they proposed to select *n* objects among the universe of web objects according to their lifetime. An object lifetime is denoted as the average time interval between the consecutive updates of the web object. As object can be stale anytime, it's necessary to fetch the fresh copy from the server to the cache. Thus prefetching objects consume extra bandwidth as it's have to download the object from the server. To reduce bandwidth consumption, it's very common tendency to fetch those objects who change less frequently.

III. Hit rate and Bandwidth Analysis

In this section, we briefly describe hit rate with prefetching and without prefetching along with the bandwidth consumption.

A. Steady state demand hit rate

According to [30], $P_{A_i}(t)$ is the probability of an object *i* being accessed within *t* time and $P_{B_i}(t)$ is the probability of no update occurred in object *i* within time *t*. So, the probability of hit is

$$P_{hit_d}(i) = \sum_i P_i \int_0^\infty P_{A_i}(t) P_{B_i}(i) dt \tag{3}$$

Suppose $P_{(a,t)}(k)$ is the probability of *k* access within time *t* with access arrival rate *a*. According to assumption of [30], request arrival follows Poisson distribution, the probability of *k* arrivals in time *t* is

$$P_{(a,t)}(k) = e^{-at} . \frac{(at)^k}{k!} \tag{4}$$

The probability of no update of object *i* within *t* time and suppose *X* = no access to object *i* within time *t* and *Y* = *k* requests in *t* and *Z* = none of the *k* requests for *i*

$$\begin{aligned} P(x) &= \sum_{k=0}^{\infty} P(Y)P(Z) \\ &= \sum_{k=0}^{\infty} \left(e^{-(at)} \frac{(at)^k}{k!} \right) (1-p_i)^k \\ &= e^{-(ap_i)t} \end{aligned}$$

which will be use in the calculation to bring out the hit rate of an object. Here probability of access to an object *i* occurring time *t* is

$$P_{A_i}(t) = (ap_i) e^{-(ap_i)t} \tag{5}$$

From [30], the lifetime of i^{th} object are exponentially distributed within average lifetime l_i, the probability of no update of an object within time *t* is

$$P_{B_i}(t) = e^{-\frac{t}{l_i}} \tag{6}$$

If we put Eq. (5) and Eq. (6) into Eq. (3), the probability of an access is hit

$$\begin{aligned} P_{hit_d}(i) &= \sum_i p_i \int_0^\infty \left((ap_i) e^{-(ap_i)t} \right) \left(e^{-\frac{t}{l_i}} \right) dt \\ &= \sum_i p_i (ap_i) \left(\frac{e^{-\left(\frac{ap_i+1}{l_i}\right)}}{-\left(\frac{ap_i+1}{l_i}\right)} \Big|_0^\infty \right) \\ &= \sum_i p_i \left(\frac{ap_i l_i}{ap_i l_i + 1} \right) \end{aligned} \tag{7}$$

This $\frac{ap_i l_i}{ap_i l_i + 1}$ represents the hit rate of the object *i* which means *i* being accessed when it is fresh. This also mentioned as freshness factor of object *i* expressed as *f*(*i*). This will help to measure the object's activity in the network.

B. Steady prefetch hit rate

They proposed to prefetch an object and keep that object always fresh if any objects $P_{goodFetch}$ is above the threshold value *T*. So, the steady hit rate in prefetching with threshold value *T* is

$$P_{hit_p}(i,T) = \sum_i p_i h_i, \text{ where } h_i = \begin{cases} 1 & P_{goodFetch>T} \\ \frac{ap_i l_i}{ap_i l_i + 1} & Otherwise \end{cases} \tag{8}$$

C. Steady State Bandwidth

They represented the steady state bandwidth for the both threshold algorithm based scheme and demand based access methods. The estimated steady state bandwidth when just demand fetches is

$$BW_{ss_d} = \sum_i s_i(1-f(i))ap_i \tag{9}$$

For threshold algorithm based prefetched, Steady state bandwidth consumed by prefetch and demand

$$BW_{ss_d} = \sum_i s_i \times h_{i,} \ \ where\ h_i = \begin{cases} \frac{1}{l_i}, & P_{goodFetch(i)} > T \\ ap_i(1-f(i)), & Otherwise \end{cases} \tag{10}$$

IV. Performance Evaluation Model

Jiang et al. [31] proposed a balance measurement of prefetching algorithm which is called *H*/*B* metric and it is defined as

$$\frac{H}{B} = \frac{\frac{Hit_{pref}}{Hit_{demand}}}{\frac{BW_{pref}}{BW_{demand}}} \tag{11}$$

Here Hit_{pref} and Hit_{demand} are the overall hit rate with and without prefetching. BW_{pref} and BW_{demand} are the total bandwidth with and without prefetching. *H*/*B* describes the ratio between hit rate and bandwidth. H/B model will help to evaluate the performance of the algorithm. Jiang et al. [31] proposed a more generalized form of *H*/*B* metric as H^k / B to give importance either on hit rate or bandwidth by controlling the value of *k*

$$\frac{H^k}{B} = \frac{(\frac{Hit_{pref}}{Hit_{demand}})^k}{\frac{BW_{pref}}{BW_{demand}}} \tag{12}$$

Here *k*>1 indicates that hit rate is preferred over the bandwidth and if bandwidth is limited then *k*<1. Jiang et al. [31] used both *H*/*B* and H^k / B metric to measure the performance of Popularity [29], Good Fetch [30], APL characteristic [31], and Lifetime [31].

V. Proposed Protocol: HitBand

In this section, we introduce our proposed model named HitBand for prefetching. HitBand is a model which prefetch objects maintaining the balance of hit rate and bandwidth with the combination of RWS where RWS is used only in the final selection process. RWS is a selection based algorithm which selects individuals according to their fitness.

RWS has been applied on genetic algorithms, course selection problem, cloud computing etc. but never used for web prefetching. According to RWS, if *ff*(*i*) is the fitness of the object *i*, the probability of this object being selected is

$$P(i) = \frac{ff(i)}{\sum_{j=1}^{N} ff(j)} \tag{13}$$

where *N* is the number of the total object in a given set S. Fitness of a Objects depends on the factor of hit rate and bandwidth. If *ff*(*i*) is the fitness function of the object *i* , hit rate of object *i* is h_i, bandwidth is b_i and wants to balance hit rate with the bandwidth then the fitness function of object *i* is

$$ff(i) = \frac{h_i}{b_i^k} \tag{14}$$

Here *k* is the bandwidth constant. If we use the definition of h_i and b_i from the Eq. (8) and Eq. (10)

$$ff(i) = \frac{\frac{ap_i l_i}{ap_i l_i + 1}}{\left(ap_i(1-f(i))s_i\right)^k} \tag{15}$$

Here *f*(*i*) is the freshness factor provided by Venkataramani et al. [30] in the Eq. (7) and if we put the value of *f*(*i*) in the Eq. (15)

$$\begin{aligned} ff(i) &= \frac{\frac{ap_i l_i}{ap_i l_i + 1}}{\left(ap_i(1-f(i))s_i\right)^k} \\ &= \frac{\frac{ap_i l_i}{ap_i l_i + 1}}{\left(ap_i\left(1-\frac{ap_i l_i}{ap_i l_i + 1}\right)s_i\right)^k} \\ &= \frac{\frac{ap_i l_i}{ap_i l_i + 1}}{\left(ap_i\left(\frac{ap_i l_i + 1 - ap_i l_i}{ap_i l_i + 1}\right)s_i\right)^k} \\ &= \frac{\frac{ap_i l_i}{ap_i l_i + 1}}{\left(ap_i\left(\frac{1}{ap_i l_i + 1}\right)s_i\right)^k} \\ &= \frac{\frac{ap_i l_i}{ap_i l_i + 1}}{\left(\frac{ap_i s_i}{ap_i l_i + 1}\right)^k} \end{aligned} \tag{16}$$

Now, if we put Eq. (16) into Eq. (13)

$$P(i) = \frac{\frac{ap_i l_i}{ap_i l_i + 1}}{\sum_{j=1}^{N} \frac{\frac{ap_j l_j}{ap_j l_j + 1}}{\left(\frac{ap_j s_j}{ap_j l_j + 1}\right)^k}} \quad (17)$$

Here k is the bandwidth constant and $0 <= k <= 1$. k depends on the bandwidth availability. Bandwidth availability creates issues on prefetching if it is not plenteous. If bandwidth is enormous, value of k will be 0. If bandwidth is limited, k will be $0 < k <= 1$ according to the bandwidth allocation. Objects with a *RWS* value calculated from Eq. (17) exceeding a certain threshold value will be selected for prefetching. This protocol is designed in the consideration of both hit rate and bandwidth. In enormous network bandwidth, it focuses only on hit rate but in mid and low network bandwidth, it focuses on hit rate along with bandwidth. Pseudo code is provided in Algorithm 1. According to Algorithm 1, S is the set of objects, n is the number of objects to select for prefetching, O is the object with property of s, p, l where s is the size of the object, p is access frequency and l is the lifetime, a is the total access rate, k is the bandwidth constant. First, it calculates hit rate of each objects. After calculating hit rate, it calculates bandwidth of each objects. After that, it generates fitness value of each objects using the Eq. (14). After calculating fitness value, it calculates total sum of fitness values of all the objects. After that, it calculates selection probability of each objects the Roulette Wheel Selection which is provided in the Eq. (17).

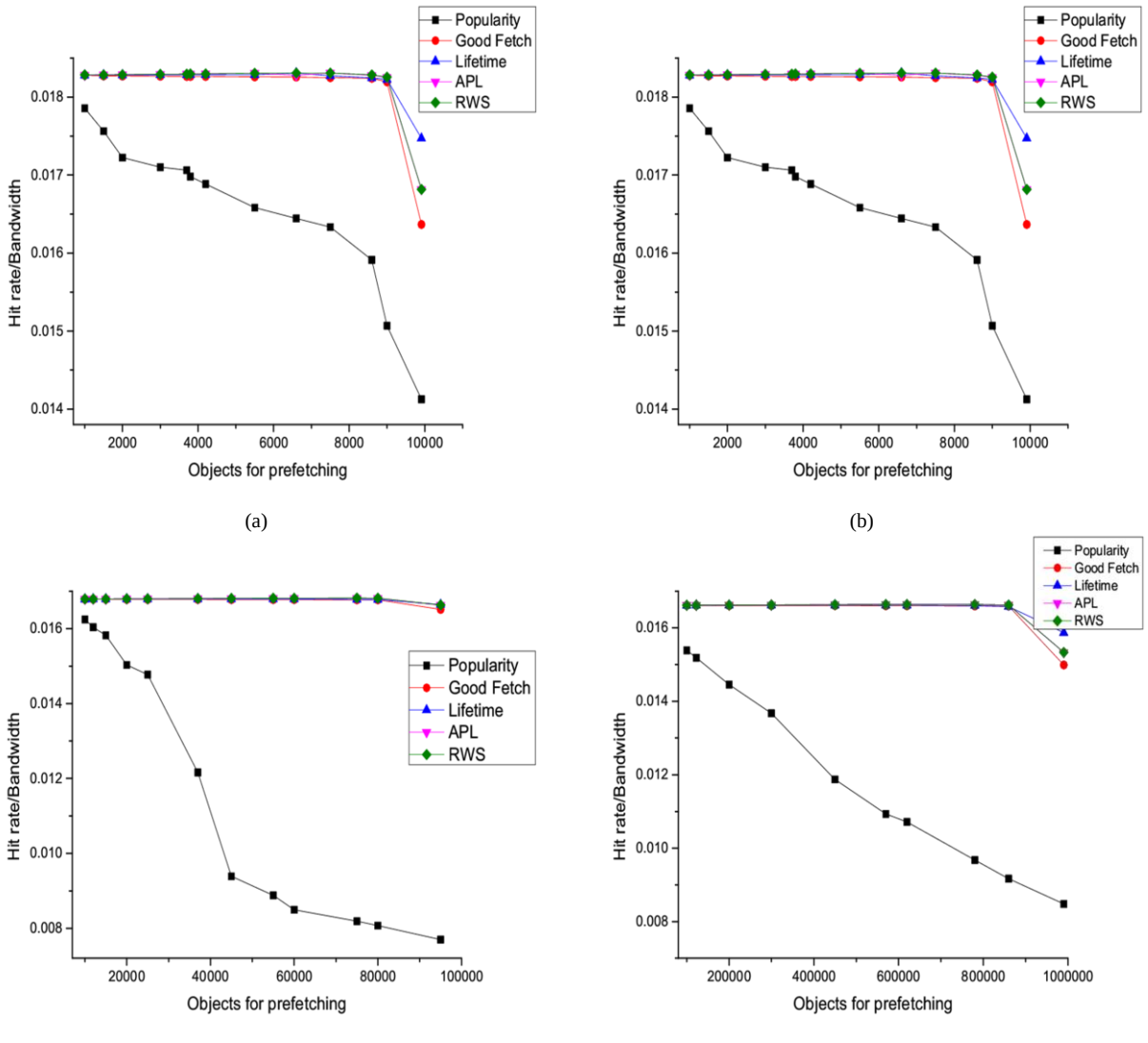

Fig.2. Hit rate/bandwidth analysis: analysis of (a) 1000 objects, (b) 10000 objects, (c) 100000 objects, (d) 1000000 objects

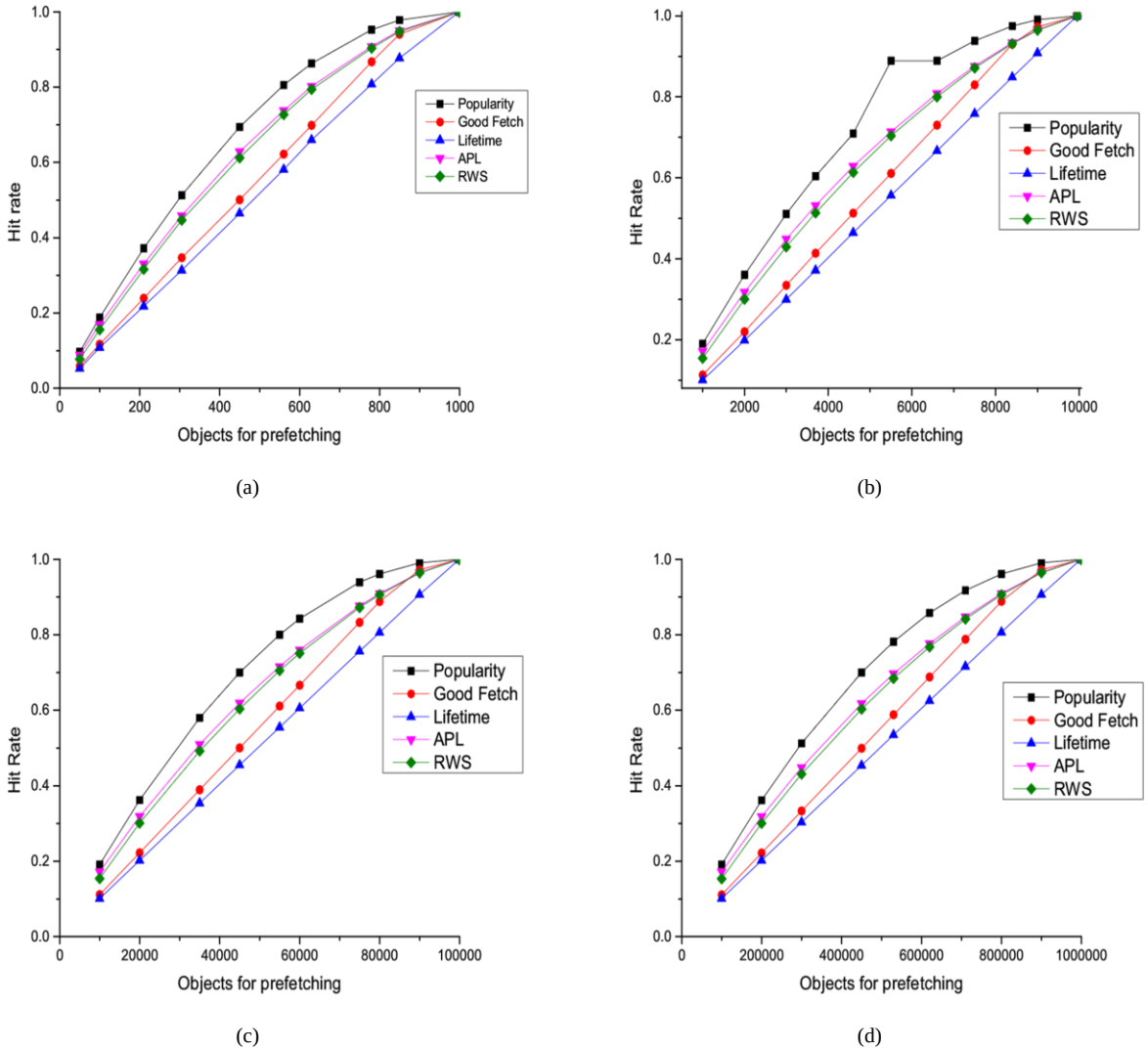

Fig.3. Hit rate analysis: Analysis of (a) 1000 objects, (b) 10000 objects, (c) 100000 objects, (d) 1000000 objects

Finally, it selects *n* objects with *RWS* value that crosses a certain threshold value. Now, we calculate both time and space complexity of our proposed protocol. To calculate h_i, it'll take *O* (1) to calculate for single object and as object is up to *S*, so the overall complexity for calculating h_i up to *S* is *O*(*S*). Same for calculating b_i, f_i, Sum and *RWS* take *O*(*S*) time. Selection of *n* objects among the *S* whose *RWS* value exceeds a certain threshold value can be done in many ways. If we use randomization in selection, it'll take *O*(*S*) to complete. Thus the overall complexity of the protocol is *O*(*S*). So selection process has also an impact on the time complexity. Space complexity depends on the space consumed by the object information (s_i, p_i, l_i where *s* is the size of the object, *p* is access frequency and *l* is the lifetime), h_i, b_i, f_i , *RWS* which will be up to *S* where *S* is the total set of web objects.

VI. Performance Evaluation and Results

We built our testing engine using JavaScript and used AngularJS framework to maintain the engine. We ran simulation on algorithms such as prefetch by popularity [29], good fetch [30], apl characteristic [31], lifetime [31] and our HitBand. For the simulation, we have used object size s_i randomly between 1 and 1000000 bytes and object's lifetime l_i between 1 and 100,000 seconds and access frequency p_i which follows *Zip*'s distribution with α=0.75 [31] and the total access rate *a*=0.01/*second*. We ran our test through three channel like *H*/*B* model, hit rate and bandwidth. We took 4 set of objects like 1000, 10000, 100000, 1000000 for testing on every channel with 5 set of algorithm.

In the testing based on H/B model, we use *k*=0.001 with a very little control of bandwidth. We used equivalent equation like Eq. (11) and Eq. (12) for H/B model. Figure 2 shows the status after performing

simulation on algorithms such as prefetch by popularity [29], good fetch [30], apl characteristic [31], lifetime [31] and our HitBand. Fig. 2a shows the analysis of the 1000 objects, Fig. 2b shows the analysis of the 10000 objects, Fig. 2c shows the analysis of the 100000 objects, and Fig. 2d shows the analysis of the 1000000 objects. From the Fig. 2, good fetch [30], APL characteristic [31], lifetime [31] and our HitBand shows almost same performance but prefetch by popularity [29] shows very poor result in H/B model.

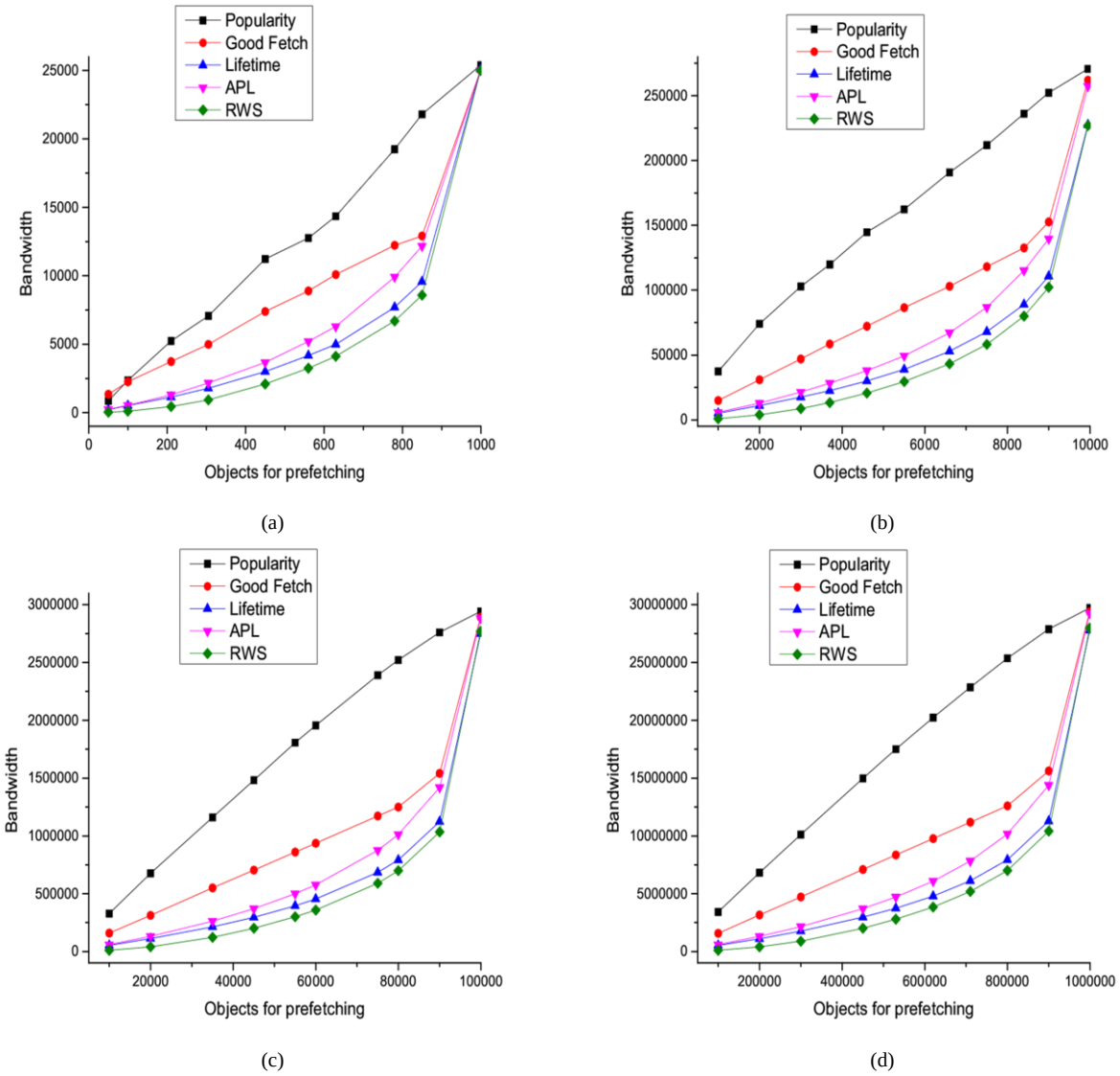

Fig.4. Bandwidth analysis: analysis of (a) 1000 objects, (b) 10000 objects, (c) 100000 objects, (d) 1000000 objects

In the testing of hit rate, we use k=0.001 with very little control of bandwidth. Figure 3 shows the status after performing simulation on algorithms such as prefetch by popularity [29], good fetch [30], APL characteristic [31], lifetime [31] and our HitBand. Figure 3a shows the analysis of the 1000 objects, Figure 3b shows the analysis of the 10000 objects, Figure 3c shows the analysis of the 100000 objects and Figure 3d shows the analysis of the 1000000 objects. From the Figure 3, Our HitBand performs better than prefetch by lifetime [31] and good fetch [30] and shows almost same performance as APL characteristic [31] but popularity [29] performs better in case of hit rate.

In the testing of bandwidth, we tested our simulation through limited bandwidth allocation. In this simulation we use k=0.5. Fig. 4 shows the status after performing simulation on algorithms such as prefetch by popularity [29], good fetch [30], APL characteristic [31], lifetime [31] and our HitBand. Fig. 4a shows the analysis of the 1000 objects, Fig. 4b shows the analysis of the 10000 objects, Fig. 4c shows the analysis of the 100000 objects and Fig. 4d shows the analysis of the 1000000 objects. In limited network bandwidth from the Fig. 4, our HitBand consumes less bandwidth than other algorithms like prefetch by popularity [29], good fetch [30], apl characteristic [31], lifetime [31]. Prefetching by popularity consume heavy bandwidth than other

algorithms. After our HitBand, prefetching by lifetime [31] consumes less bandwidth than other algorithms.

Algorithm 1: Get n prefetch object

```
procedure RWS
S : Set of Web Objects
n : Number of object to fetch
O < si, pi, li >: web object
a : access rate
k : badwidth constant

i ← 0
while i < length(S) do
 h_i ← ap_i l_i / (ap_i l_i + 1)
 i ← i + 1
end while

i ← 0
while i < length(S) do
 b_i ← ap_i s_i / (ap_i l_i + 1)
 i ← i + 1
end while

i ← 0
while i < length(S) do
 f_i ← h_i / b_i^k
 i ← i + 1
end while

Sum ← 0
i ← 0
while i < length(S) do
 Sum ← Sum + f_i
 i ← i + 1
end while

i ← 0
while i < length(S) do
 RWS_i ← f_i / sum
 i ← i + 1
end while

Select n object with maxium RWS
return selected objects
end procedure
```

VII. Conclusions

We have designed, developed and evaluated HitBand, a prefetching model whose goal is to increase hit rate with the consideration of available bandwidth. The basic difference between Hitband and generic algorithms (prefetch by popularity [29], good fetch [30], lifetime [31], apl characteristic [31]) is that generic algorithms focus on either hit rate or bandwidth but HitBand considers both. We designed our HitBand with the combination of RWS. In this paper, we not only proposed our HitBand but also provided data analysis comparison with the generic algorithms like prefetch by popularity [29], good fetch [30], apl characteristic [31], lifetime [31]. We have tested our HitBand both in limited bandwidth and large bandwidth. We have run simulation on HitBand along with other existing algorithms through H/B model, hit rate and bandwidth. In the scenario of low bandwidth, our HitBand consumes less bandwidth than other algorithm. In the scenario of hit rate and H/B model, our HitBand shows significant results. Along with the other algorithms, our HitBand is very easy to implement and it can adjust easily in any scenario.

References

[1] W. Help, FAQ, What is the difference between the web and the internet?, W3C. 2009. (2015).

[2] S. Sulaiman, Siti, A. Abraham, S. Sulaiman, Web caching and prefetching: What, why, and how?, IEEE (2008) 1–8 doi:10.1109/ITSIM.2008.4631949.

[3] J. Wang, A survey of web caching schemes for the internet, ACM Computer Communication Review 25(9) (1999) 36–46. doi:10.1145/505696.505701.

[4] Cisco, Cisco visual networking index: Forecast and methodology (2013) 2012–2017.

[5] L. Atzori, A. Iera, G. Morabito, The internet of things: A survey, Computer Networks 54(15) (2010) 2787–2805. doi:10.1016/j.comnet.2010.05.010.

[6] A. Al-Fuqaha, Internet of things: A survey on enabling technologies, protocols, and applications, IEEE Communications Surveys and Tutorials (Volume: 17, Issue: 4) (2015) 2347–2376 doi:10.1109/COMST.2015.2444095.

[7] Z. M, S. H, N. M., Understanding and reducing web delays, IEEE Computer Magazine 34(12) (2001) 30–37.

[8] A. C, W. J. L, Y. P. S., Caching on the world wide web, IEEE Trans. Knowledge and Data Engineering 11(1) (1999) 94–107.

[9] B. D. Davison, A web caching primer, IEEE Internet Computing 5 (2001) 38–45. doi:10.1109/4236.939449.

[10] F. A, C. R, D. F, G. G, R. M., Performance of web proxy caching in heterogeneous bandwidth environments, In Proc. the IEEE Infocom'99 Conference (1999) 107–116 doi:10.1109/INFCOM.1999.749258.

[11] W. Ali, S. M. Shamsuddin, A. S. Ismail, A survey of web caching and prefetching, Int. J. Advance. Soft Comput. Appl. 3(1) (2011) 18–44.

[12] H. Chen, Pre-fetching and re-fetching in web caching systems: Algorithms and simulation, Master Thesis, TRENT UNIVESITY, Peterborough, Ontario, Canada (2008).

[13] T. Chen, Obtaining the optimal cache document replacement policy for the caching system of an ec website, European Journal of Operational Research. 181(2) (2007) 828. doi:10.1016/j.ejor.2006.05.034.

[14] Y. Ma, X. Liu, S. Zhang, R. Xiang, Y. Liu, T. Xie, Measurement and analysis of mobile web cache performance, WWW '15 Proceedings of the 24th International Conference on World Wide Web (2015) 691–701 doi:10.1145/2736277.2741114.

[15] G. G. Vijayan, J. J. S., A survey on web pre-fetching and web caching techniques in a mobile environment, The First International Conference on Information Technology Convergence and Services (2012) 119–136 doi:10.5121/csit.2012.2111.

[16] T. M. Kroeger, D. D. E. Long, J. C. Mogul, Exploring the bounds of web latency reduction from caching and prefetching, Proceedings of the USENIX Symposium on Internet Technologies and Systems on USENIX Symposium (1997) 2–2.

[17] M. Abrams, C. R. Standridge, G. Abdulla, S. Williams, E. A. Fox, Caching proxies: limitations and potentials, Proceedings of the 4th International WWW Conference, Boston (1995).
[18] H. Lee, B. An, , E. Kim, Adaptive prefetching scheme using web log mining in cluster-based web systems, 2009 IEEE International Conference on Web Services (ICWS) (2009) 903–910.
[19] A. Abhari, S. P. Dandamudi, S. Majumdar, Web object-based storage management in proxy caches, Future Generation Computer Systems Journal 22(1-2) (2006) 16–31. doi:10.1016/j.future.2005.08.003.
[20] L. Jianhui, X. Tianshu, Y. Chao, Research on web cache prediction recommend mechanism based on usage pattern, First International Workshop on Knowledge Discovery and Data Mining(WKDD) (2008) 473–476 doi:10.1109/WKDD.2008.9.
[21] D. Kumar, R. Patel, An efficient approach for optimal prefetching to reduce web access latency, International Journal of Scientific and Technology Research (2014) 3(7).
[22] V. Sathiyamoorthi, V. M. Bhaskaran, Optimizing the web cache performance by clustering based pre-fetching technique using modified art, International Journal of Computer Applications 44(1) (2012) 7–9. doi:10.5120/6225-8190.
[23] G. Pallis, A. Vakali, J.Pokorny, A clustering-based prefetching scheme on a web cache environment, Computers and Electrical Engineering 34(4) (2008) 309–323. doi:10.1016/j.compeleceng.2007.04.002.
[24] W. Feng, S. Man, G. Hu, Markov tree prediction on web cache prefetching, Software Engineering, Artificial Intelligence(SCI), SpringerVerlag Berlin Heidelberg, 209 (2009) 105–120 doi:10.1007/978-3-642-01203-7\s\do5(9).
[25] S. Gawade, H. Gupta, Review of algorithms for web pre-fetching and caching, International Journal of Advanced Research in Computer and Communication Engineering Vol. 1, Issue 2, April 2012.
[26] R. Kaur, V. Kiran, Various techniques of web pre-fetching, International Journal of Advanced Research in Computer Science and Software Engineering Volume 4, Issue 11, November 2014.
[27] H. J., Adaptation in natural and artificial systems, University of Michigan Press, Ann Arbor (1975) doi:10.1137/1018105.
[28] Wikipedia, Fitness proportionate selection, https://en.wikipedia.org/wiki/Fitnessṗroportionatėselection
[29] E. Markatos, C. Chironaki, A top 10 approach for prefetching the web, Proc. INET98: Internet Global Summit (1998).
[30] A. Venkataramani, P. Yalagandula, R. Kokku, S. Sharif, M. Dahlin, The potential costs and benefits of longterm prefetching, Computer Communications 25(4) (2002) 367–375. doi:10.1016/S0140-3664(01)00408-X.
[31] Y. Jiang, M. Wu, W. Shu, Web prefetching: Cost, benefits and performance, 11th World Wide Web Conference (WWW) (2002).
[32] L. Breslau, P. Cao, L. Fan, G. Philips, S. Shenker, Web caching and zipf-like distributions: Evidence and implications, Proc. IEEE Infocom 1 (1999) 126–134. doi:10.1109/INFCOM.1999.749260.
[33] C. Cunha, A. Bestavros, M. Crovella, Characteristics of www client-based traces, Technical Report TR-95-010, Boston University, CS Dept., Boston (1995).
[34] N. Nishikawa, T. Hosokawa, Y. Mori, K. Yoshidab, H. Tsujia, Memory based architecture with distributed www caching proxy, Computer Networks 30 (1–7) (1998) 205–214. doi:10.1016/S0169-7552(98)00117-2.
[35] M. Crovella, A. Bestavros, Self-similarity in World Wide Web traffic: Evidence and possible causes, IEEE/ACM Trans. on Networking 5(6) (1997) 835–746. doi:10.1109/90.650143.
[36] M. Crovella, P. Barford, The network effects of prefetching, Proc. IEEE Infocom (1998) 1232–1239 doi:10.1109/INFCOM.1998.662937.
[37] Nashaat el-Khameesy, Hossam Abdel Rahman Mohamed, A Proposed Model for Web Proxy Caching Techniques to Improve Computer Networks Performance, I.J. Information Technology and Computer Science 5(11) (2013) 42-53. doi: 10.5815/ijitcs.2013.11.05.

Authors' Profiles

Islam Anik received his Bachelor degree in Software Engineering in 2014 from American International University of Bangladesh (AIUB). In 2012, he joined Bengal Solutions Ltd. as a software engineer. In January 2014, he joined dynamic software ltd. In August 2014, he joined Next IT Ltd. as a software engineer. Currently, he is doing Masters in Computer science at American International University of Bangladesh (AIUB). His research interest includes distributed system, internet of things (IoT), web of things (WoT), big data, data science, intelligent system and semantic web.

Akter Arifa received her Bachelor degree in Computer Science and Software Engineering in 2014 from American International University of Bangladesh (AIUB). In 2014, she joined Rupshi Sweaters Ltd. as IT Manager. Currently, she is doing Masters in Computer science at American International University of Bangladesh (AIUB). Her research interest includes distributed system, data mining, big data, intelligent system and semantic web.

Hamid Md. Abdul received his Bachelor of Engineering degree in Computer & Information Engineering in 2001 from International Islamic University Malaysia (IIUM). In 2002, he joined as a lecturer in the Computer Science & Engineering department, Asian University of Bangladesh, Dhaka. He received the Ph.D. degree from Kyung Hee University, South Korea in August 2009 from the Computer Engineering department. In September 2009, he joined as a faculty member in the department of Information & Communications Engineering at Hankuk University of Foreign Studies (HUFS), South Korea. In Septemer 2012, he joined Green University of Bangladesh (GUB) as an Assistant professor and chairmen and held that position to May 2013. He had been a faculty member in the Department of Computer Engineering, College of Computer Science & Engineering, Taibah University, Madinah, KSA. Currently he holds a faculty position in the Department of Computer Science at American International University-Bangladesh. He is the TPC member of TNS-2011 and ICCIT-

2011, and member of KSII (Korean Society for Internet Information). His research interest includes distributed systems, wireless sensor, mesh, ad hoc, and opportunistic networks with particular emphasis on network security, reliability, fairness, and quality of service (QoS) issues

How to cite this paper: Islam Anik, Akter Arifa, Hamid Md. Abdul,"HitBand: A Prefetching Model to Increase Hit Rate and Reduce Bandwidth Consumption", International Journal of Information Engineering and Electronic Business(IJIEEB), Vol.9, No.1, pp.36-46, 2017. DOI: 10.5815/ijieeb.2017.01.05

