International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-9 Issue-2, December, 2019

Neural Network Based Prefetching Control Mechanism

Sonia Setia, Jyoti, Neelam Duhan

Abstract**: *An important issue incurred by users that limits the use of internet is the long web access delays. Most efficient way to solve this problem is to use "Prefetching". This paper is an attempt to dynamically monitor the network bandwidth for which a neural network-based model has been worked upon. Prefetching is an effective and efficient technique for reducing users perceived latency. It is a technique that predicts & fetches the web pages in advance corresponding to the clients' request, that will be accessed in future. Generally, this prediction is based on the historical information that the sever maintains for each web page it serves in a chronological order. This is a speculative technique where if predictions are incorrect then prefetching adds extra traffic to the network, which is seriously negating the network performance. Therefore, there is critical need of a mechanism that could analyze the network bandwidth of the system before prefetching is done. Based on network conditions, this model not only guides if the prefetching should be done or not but also tells number of pages which are to be prefetched in advance so that network bandwidth can be effectively utilized. Proposed control mechanism has been validated using NS-2 simulator and thus various adverse effects of prefetching in terms of response time and bandwidth utilization have been reduced.***

Keywords**: *Network Bandwidth, Neural Network, Prediction, Prefetching***

I. INTRODUCTION

Due to enormous information present on the World Wide Web, users have been experiencing long delays while accessing files from World Wide Web. Prefetching is the solution to render these delays. The intent behind prefetching is to take benefit of the idle time between two network accesses i.e. when users are viewing the web pages which are just downloaded. In this idle period, prefetching estimates and fetches the additional web pages which will be accessed in near future based on some intelligence added to the applications so that users' waiting time can be reduced and thus experience of using Internet could be improved. If the prefetched web pages are indeed requested, these can be accessed with negligible delay. If the system could exactly predict those web pages which a user will request next, we will fetch only those web pages in advance and user will enjoy zero latency. Unfortunately, some prefetched web pages may never be used which results in wastage of network bandwidth and adds to the principal cost of prefetching. In literature, there are a lot of prefetching techniques discussing prediction algorithms, their accuracy, precision and hit ratio etc. which are mainly its host aspects. Second aspect is networking aspect of the prefetching i.e. how to determine the number of web pages to prefetch to reduce its adverse effects on the network. Though, prefetching is taking advantage of users' idle time, however, it is also necessary to consider whether network is idle at prefetching time or not.

Based on these two aspects, prefetching scheme basically consists of two modules:

A. *Prediction Module*

After a users' current request is satisfied, prediction module immediately starts working and predicts the future requests of the user by computing the probability with which the web pages will be accessed in near future. Different types of prediction algorithms have been used in literature for this module.

B. *Threshold Module*

Based on network conditions, this module takes decision for Prefetching. If it allows for prefetching then it computes value of prefetching threshold i.e. how many numbers of documents which are to be prefetched to achieve optimum performance. This module is independent of the prediction module i.e. same threshold algorithm can be applicable with different prediction algorithms.

This paper focused on second aspect of prefetching i.e. Threshold module which determines the prefetch threshold based on network conditions in real time. In this view, a control mechanism has been proposed which uses the ping's ICMP (Internet control message protocol) messages to compute the RTT (round trip time) and network bandwidth is also measured to control the prefetch threshold so that network performance can be optimized. It employs a Neural Network model over the RTT and Network Bandwidth basis which it tells if the system is ready for prefetching or not and if yes, how many web pages to be prefetched to optimize the network usage.

The remainder of the paper is organized as follows. Brief review of literature work has been given in section 2. Section 3 presented the proposed work in which an algorithm has been developed to determine the prefetch threshold based on network conditions.

Revised Manuscript Received on December 30, 2019.

* Correspondence Author
Sonia Setia, Computer Science, YMCA, Faridabad, India. Email: setiasonia53@gmail.com
Jyoti, Computer Science, YMCA, Faridabad, India. Email: justjyoti.verma@gmail.com
Neelam Duhan, Computer Science, YMCA, Faridabad, India. Email: neelam.duhan@gmail.com

In section 4, summarized results of evaluation of the proposed work through trace-driven simulations have been shown. Finally, conclusion has been presented in section 5.

II. RELATED wORK

Despite the benefits of prefetching, it can increase the network traffic if not employed in a controlled way. An aggressive prefetching can prefetch many extra objects which are never requested by user in future. This can severally decrease the system's performance. In literature, few approaches have been proposed to control the adverse effects of prefetching.

R. Chen et al. [1] proposed a cache optimization method to reduce the network traffic usage which utilized data mining technique with clustering concept. By gathering the current feedback data from SPN (Smart Protect Network (SPN)), authors were able to cluster the data in groups based on their similarity, and by deploying these data to client side, authors achieved the reduction of traffic usage. In prototyping, this design for military communications reduced network traffic by more than 20% and the speed of file scanning time increased by 12%. Authors tried to reduce the network traffic by optimizing the cache but it does not help in prefetching.

J. Domenech et al. [2] proposed an intelligent prefetching mechanism which dynamically adjusted the aggressiveness of prefetching at server side. Authors calculated the extra traffic generated by prefetching based on the type of requests known by the server i.e. prefetch request/user requests not prefetched. They developed traffic estimation model using prefetch rate metric which is based on prefetch hits and prefetch misses. Authors tried to reduce the extra traffic based on the performance of Prefetching technique but they didn't consider the network conditions.

Pingshan liu et al. [3] proposed a prefetching strategy for peer to peer video on demand system, to offload the servers effectively. In this paper, authors calculated the server load by using exponential weighted moving average approach periodically. Based on this, authors determined which segment of a movie a peer should prefetch. Authors tried to reduce the server load based only on the weighted moving average approach but they too did not consider network conditions.

Z. chen. et al. [4] derived a centralized solution for minimum departure misses problem. Due to peer departure, some chunks only hosted on these peers disappear in the system. They examined how to allocate extra bandwidth to decrease departure misses in peer to peer video on demand. In addition, they also proposed a predictor-based bandwidth allocation algorithm that reduced departure misses problem through service differentiation. Authors tried to reduce the departure misses problem by allocating extra bandwidth but not considered network conditions.

Divya E et al. [5] proposed an approach to reduce the server load by using peer to peer network with caching & replication. The proposed system focused on hybrid caching including cache prefetching and opportunistic cache update. In addition, system has been further improved by adding replication capability to peers. Authors reduced the server load by using replication of data which is not a good solution.

A Bestavros et al. [6] presented two server-initiated protocols to improve the performance of World Wide Web. First protocol is for a hierarchical data dissemination mechanism which is based on geographic and temporal locality of reference properties exhibited in client access patterns. Geographical locality of reference means accessed objects are likely to be accessed again later on by 'nearby' clients. Temporal locality of reference means frequently accessed objects are likely to be accessed in near future. Second protocol employed speculative service which means a user's request is served by server by sending the requested document, in addition a number of other documents that are going to be requested in near future. This speculation reduced service time by exploiting the spatial locality of reference property which implies that an object neighboring a recently accessed object is likely to be accessed again later on. Authors reduced the user's service time by exploiting geographical, temporal and spatial locality but they did not consider the factors like system load and network conditions which can greatly affect the network performance.

- Most of the work in literature on prefetch uses a fixed prefetch threshold i.e. a fixed number of web pages to prefetch.
- The problem with these approaches is that they do not consider either system load or network conditions which can negatively impact the network performance.

However, there should be more constraints on prefetching when network condition is severe. In this work, the above-mentioned problems have been resolved by computing the dynamic prefetch threshold based on network conditions.

III. PROPOSED WORK

Our prefetching technique optimizes the trade-off between latency and system resource usage (network link, server etc.). It is done by predicting which web pages are likely to be accessed in future and choosing only some of them to prefetch to optimize the network performance. The first task is accomplished by prediction module which can use any of the prediction algorithms in literature and second one by threshold module which is the main focus of our work in which we evaluate the degree to which prefetching must be effective for both cases i.e. (latency and resource usage). Also, since this threshold module is independent of the Prefetch module, it can be easily applicable with any of the Prediction module.

To optimize the trade-off between resource usage and latency, a prefetch threshold-based control mechanism has been proposed. It is the integration of the ping RTT and network bandwidth measurement to estimate the network performance so that it can control the prefetching.

Ping [1] is an ICMP echo message used to show the Round-Trip Time (RTT) with some additional information such as max/min/avg RTT, number of packets sent and received and packet drops. Round Trip Time [2] is the time measured in milliseconds required to get response corresponding to a request RTT is typically measured using a ping. Ping RTT has been employed here because:

- It is being supported widely.
- Do not interface with host aspects.

In addition, bandwidth is also introduced to control mechanism as bandwidth measurement tools are becoming mature these days.

There are a lot of tools available for network bandwidth measurement such as Bprobe, Nettimer, Pathrate, Sprobe, Pathchar, Pchar and Pathhead [3]. Bandwidth represents the amount of data that network can transfer per unit of time.

RTT has been chosen to optimize the network performance while prefetching because it is easy to get the value of RTT by using PING messages and low value of RTT indicates the good indication of network. Current value of RTT can be taken from the average of latest three measurements. In addition, bandwidth has also been considered to check the network conditions because sometimes, even with the high value of RTT, network could still be in good condition. Therefore, only RTT is not sufficient to estimate the network conditions. Based on the network conditions estimated through Ping RTT and network bandwidth, control mechanism decides whether to prefetch or not which is based on the algorithm given below:

- Algorithm 1: Network condition-based control mechanism

```
Input: Current round trip time($R_{cur}$), Current Bandwidth($BW_{cur}$)
Output: Prefetch Threshold(PT)
Begin
Choose threshold value of round trip time ($R_{th}$)and bandwidth ($BW_{th}$)
Read round trip time ($R_{cur}$) and Bandwidth($BW_{cur}$)
If $R_{cur} < R_{th} || BW_{cur} > BW_{th}$
Set Prefetch ON
else
Set Prefetch Off
End
```

In this algorithm, firstly, threshold value for RTT and bandwidth has been chosen basis the number of performed experiments. For normal traffic, average RTT observed in a study[15] is 168.9 msec. Then, it reads the current RTT if it is less than threshold RTT then it allows prefetching. Otherwise it checks on network bandwidth if current bandwidth is greater than average bandwidth then also it allows prefetch otherwise it inhibit prefetching.

Here, major implementation issue was to build up a threshold module that has the ability of self-learning. There are various methodologies available for this issue. One of them is Neural Networks. The Neural Network methodology has been around since late 1950s and came into practical use for all-purpose applications since mid-1980s. Because of its flexibility against distortions in the input data and its ability of self-learning, neural network methodology is often good at answering problems which cannot be solved algorithmically [14]. Based on the discussed control algorithm, Threshold module employs Neural Network to optimize the network performance which takes RTT and network bandwidth as input. Main advantage of using Neural Network here is that it could not only make predictions but it is also able to adjust itself according to the situation.

In the next subsection, we present neural network basics and neural network based proposed model.

A. *Neural Network*

A Neural Network is a collection of artificial neurons. Fig. 1 represents architecture of a simple NN that has been used for this paper. It contains an input, output and one hidden layer. Nodes of input layer are connected to the nodes of hidden layer and nodes of hidden layer are connected to the nodes of output layer. Initially, random weights have been assigned to every connection of the network which are adjusted during network training. Input layer represents raw information which is fed to the network. In our case, it is set of RTT and Network Bandwidth. The advantage of using Hidden layer is that it permits neural network to develop its own representation and specifies the network condition whether network condition is severe or normal. Output layer receives information from the hidden layer and after processing on it, produces an output whether to prefetch or not. The output from the neuron is computed by using the Activation Function. The purpose of the activation function is to introduce non-linearity into the output of a neuron. Because most real-world data are nonlinear so, neurons must learn these nonlinear representations.

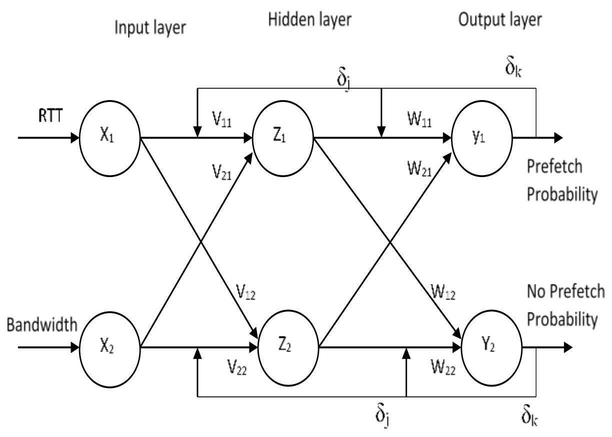

Fig. 1.Graphical representation of neural network

Every activation function takes a single number and performs mathematical operation on it as presented by J. Márquez et al.[2]. Following are major activation functions used in literature:

- Sigmoid: takes a real-valued input and compresses it to range between 0 and 1

  $\sigma(x) = 1 / (1 + \exp(-x))$

- tanh: takes a real-valued input and compresses it to the range [-1, 1]

  $\tanh(x) = 2\sigma(2x) - 1$

- ReLU: ReLU stands for Rectified Linear Unit. It takes a real-valued input and thresholds it at zero means replaces negative values with zero.

  $f(x) = \max(0, x)$

The activation function is used to turn an unbounded input into a fine predictable output. Sigmoid function is commonly used in literature. The sigmoid function outputs in the range of (0,1) means compress $(-\infty, +\infty)$ to (0,1) i.e. big negative numbers become ~0, and big positive numbers become ~1.

For classification, it is typical for the output to be a sigmoid function of its inputs (because there is no point in predicting a value outside of [0,1]). Therefore, in this work, we are using sigmoid function to classify the input either in ‘Prefetch’ or in ‘No Prefetch’.

Further, in proposed work, Neural Network has been trained using ‘Backpropagation’ training algorithm. During training, the input vector is fed to the input layer, after which it spreads through the network from layer to layer. As a result, output signals are generated corresponding to the provided input. The intent behind the ‘Backpropagation’ algorithm is pretty simple i.e. output of network is evaluated against targeted output. Weights to the connection are modified and outputs are calculated in repetitive manner until error is small enough to be ignored.

In Backpropagation training algorithm, training process of the network involves two passes- forward pass and backward pass. In the forward pass, outputs are evaluated and compared with targeted outputs. Then, errors from targeted and actual outputs are calculated. During the forward pass, all weights to the connections of the network are fixed. During the backward pass, all weights are adjusted according to the error correction rule. Forward and backward passes are repeated until the error is low enough.

Backpropagation algorithm finds the minimum value of error function in weight space using a technique called delta rule or gradient descent. The weights that minimize the error function are finally considered to be a solution for the given problem. Concept of Backpropagation is illustrated in Fig. 2

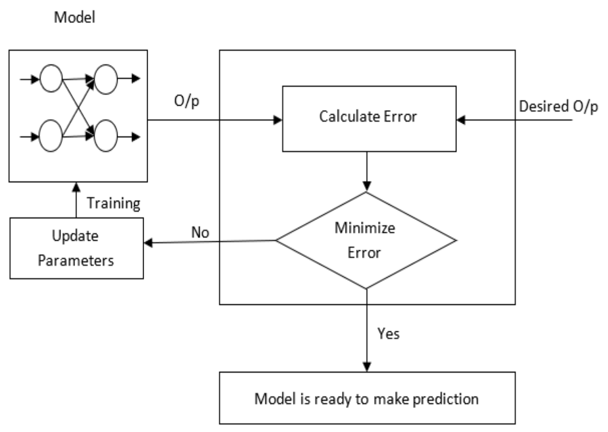

Fig. 2.Concept of Backpropagation

Here, with the help of Backpropagation training algorithm, Neural Network model has been proposed which takes RTT and Bandwidth as its input to check the network conditions and then classify them into two categories i.e.

- Prefetch: Prefetching should be done
- No Prefetch: Prefetching should not be done

The proposed Neural Network model takes the desired output which has been calculated by using Algorithm 1. Basis Algorithm 1, our model is validated. Initially, it initializes the maximum permissible error and learning rate. Learning rate is a hyper-parameter which controls how much weights are updated during training process. Its value varies between 0-1. The simplest learning rate schedule is to decrease the learning rate linearly from a large initial value to a small value. This allows large weight updations in the beginning of training and small or fine updations in the end of training. The training algorithm for the proposed Neural network model is as follows:

- Algorithm 2: Neural network model

Step 1: Input rtt and bandwidth to the network are defined as follows:

$[x_1 \quad x_2]$

Step 2: Initialize learning rate (α) and max. error=0.25

Step 3: Assign weights to the hidden layer and output layer (initially random small values are taken in the range of 0-1)

Step 4: For each hidden unit z_j, j=1,2,..p, calculate net input of each hidden layer

$(Z_{in})_j = \sum_{i=1}^{n} x_i v_{ij}$

Step 5: Calculate hidden layer output using sigmoidal activation function over $(Z_{in})_j$

$Z_j = f((Z_{in})_j)$

This will become the input for output layer.

Step 6: For each output unit y_k, k=1,2,…m, calculate net input

$(y_{in})_k = \sum_{j=1}^{p} z_j w_{jk}$

Step 7: Find the results of output layer by using sigmoidal activation function

$Y_k = f((y_{in})_k)$

Step 8: For each output unit y_k, k=1,2,..m, get a target value w.r.t. the input training pattern and computes the delta error

$\delta_k = (t_k - y_k)\, f'((y_{in})_k)$

Step 9: If calculated error δ_k is greater than max. error goto step 10 for weight updation otherwise goto step 16.

Step 10: On the basis of calculated delta error δ_k, update weights of output layer

$\Delta w_{jk} = \alpha \ \delta_k z_j$

Step 11: Also, send δ_k backward to the hidden layer

$(\delta_{in})_j = \delta_k w_{jk}$ And

$\delta_j = (\delta_{in})_j\, f'((z_{in})_j)$

Step 12: On the basis of calculated delta error δ_j, update weights of hidden layer

$\Delta v_{ij} = \alpha \ \delta_j x_i$

Step 13: Based on the change in weights, update weights for output layer units

$W_{jk}(new) = w_{jk}(old) + \Delta w_{jk}$

Step 14: Based on the change in weights, update weights for hidden layer units

$v_{ij}(new) = v_{ij}(old) + \Delta v_{ij}$

Step 15: Goto step 4 with new weights and repeat the process until error is permissible.

Step 16: return o/p in the form of Prefetch or No Prefetch.

Flow diagram depicted in Fig. 3 demonstrates the complete flow of this neural network model.

The uniqueness of this Threshold module is that it is a generalized model. As a result, it can be picked up by any of the prediction modules which employ different techniques available in literature. Our threshold module proposes a control mechanism that helps in optimizing the network load by determining if prefetching should be done or not.

Then, it computes Prefetch Threshold i.e. how many pages should be prefetched which is equal to 'T/PR' where T is the time between the end of previous request and the next request, P is the number of packets necessary to transfer a web page, R is the round trip time between client and server. According to http archive [16], average web page size is 3MB.

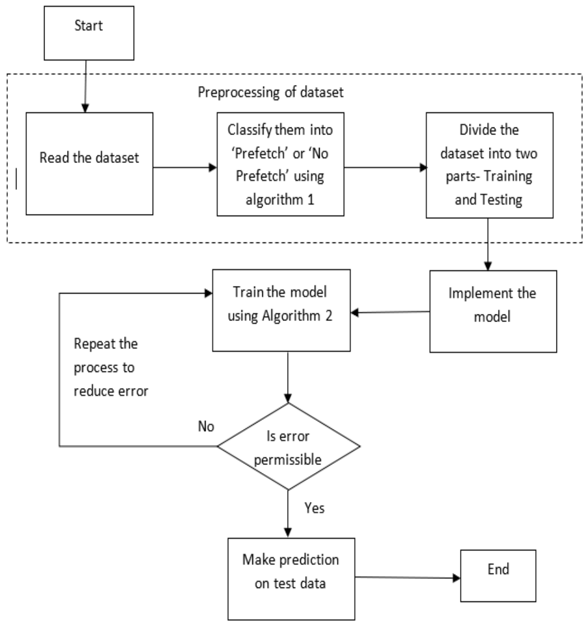
Start
Preprocessing of dataset
Read the dataset
Classify them into 'Prefetch' or 'No Prefetch' using algorithm 1
Divide the dataset into two parts- Training and Testing
Implement the model
Train the model using Algorithm 2
Repeat the process to reduce error
No
Is error permissible
Yes
Make prediction on test data
End

Fig. 3.Flow diagram for Neural Network model

- Algorithm 3: Neural Network based Prefetch Threshold Control Mechanism

```
Input: RTT, Bandwidth
Output: Prefetch Threshold (PT)
Begin
O/p ←Apply Neural Network model on the I/p patterns
If(O/p==Prefetch)
PT= T/P*R
Else
PT=0
End
```

This way, the proposed Prefetching control mechanism controls the adverse effects of prefetching. In severe network conditions, it will not allow for prefetching and in normal conditions also, it decides how much prefetching should be done based on the network conditions. Thus, network bandwidth can be effectively utilized because if traffic is too less, it can allow more prefetching of web pages to be done otherwise less prefetching. Thereby, considerably reducing the network load.

IV. EXPERIMENTAL RESULTS

As the World Wide Web is basically a dynamic environment, its fluctuating network traffic and server-load generate experimental evaluation challenge. Usually, two major evaluation methodologies are available to describe user access behaviour for Prefetching:

- Simulation based on user access traces and
- Parallel evaluation with real-time access.

In this paper, trace based simulation has been opted for evaluation because it works upon a parameterized testing environment with variable network traffic and workload. By doing so, prefetching in different network conditions can be analysed and it is also a common approach to receive repeatable effects.

The proposed control mechanism has been implemented using NS-2 network simulator [10], where for simulations of TCP, both interactive sources and bulk data are available. Additionally, it also includes a Web Cache component which is required for our experiments. NS-2 simulator is quite flexible because new traffic models can be easily added to it. Most important to us, there is a Ping Agent in NS-2 framework. We added a variable 'rtt' in the source code that stores the latest RTT value in the Ping Agent class. Ping calls are generated between two nodes every l ms. Therefore, we can get current RTT value at any time. In addition, available network bandwidth has been obtained through analyzing the trace file [11] generated by the simulator which is used for viewing network simulation traces and real-world packet trace data. We first calculate the bandwidth consumed in every 10 seconds. Then, available bandwidth can be obtained by subtracting the bottleneck bandwidth by the consumed bandwidth.

To run any simulation, we first adjust the RTT value in the simulation to a realistic level. To test the efficiency of the algorithm we adjusted the network load using Pareto models [12] by generating background traffic. We used AOL search logs trace to get a multi-user and multi-server accesses pattern which are collected for a period of three months spanning from 01 March 2006 to 31 May 2006. This collection consists of 20M web queries collected from 650k users over a three-month period. The dataset is divided into two sub-sets, one for training and another for testing in the proportion of 80:20. Training set has been used to train the neural network while testing set comprising of various input sets which has been used to run multiple test cases. We have implemented an improved prediction algorithm developed by Setia et al. in [13]. We run the simulations over a 2.67 Ghz Intel Core i5 processor with 4 GB RAM running the Windows operating system. To test and validate our control mechanism multiple testing experiments were run using our proposal having three main agendas 'without prefetch', 'prefetch', 'controlled prefetch'. From Table 1 and 2, we can observe that prefetch has great effect on the network and controlled prefetch method reduces the adverse effect of prefetching on RTT as well as bandwidth utilization compared to prefetch without control.

It is because controlled prefetching avoids a large number of retransmissions due to packet timeouts when RTT is high.

Table 1: RTT comparison (time in ms)

Without Prefetch	Prefetch	Controlled Prefetch	Improvement
123.0287	144.424	127.435	13%
103.744	137.456	110.3478	24%
107.0043	133.056	115.0435	15%
113.786	142.5643	121.0346	17%
101.0056	128.056	113.6732	13%

Table 2: Bandwidth utilization comparison (in KB/s)

Without Prefetch	Prefetch	Controlled Prefetch	Improvement
1023.287	1044.424	1027.435	1.6%
1014.744	1037.456	1020.348	1.7%
987.043	1023.056	1005.045	1.8%
1031.786	1072.564	1045.036	2.5%
1042.005	1108.056	1063.672	4.2%

V. CONCLUSION

This paper proposed a prefetch threshold-based control mechanism to prefetching. The integration of ping RTT and network bandwidth measurements has been used to determine the condition of network at the time of usage. In addition, an algorithm has been developed to compute the dynamic prefetch threshold based on network conditions. Neural network-based model has been deployed to check the network conditions i.e. whether it is appropriate for prefetching or not. By employing the proposed mechanism,

- The overall prefetch system performance is improved by utilizing the available bandwidth effectively without overloading the network.
- As a result, trade-off between user's waiting time and bandwidth usage has been optimized.

REFERENCES

1. R. Chen et al., "Cache Optimization Method to Reduce Network Traffic in Communication Systems," 2018 9th International Symposium on Parallel Architectures, Algorithms and Programming (PAAP), Taipei, Taiwan, 2018, pp. 122-125.
2. J. Márquez, J. Domènech, J. A. Gil and A. Pont, "An Intelligent Technique for Controlling Web Prefetching Costs at the Server Side," 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, Sydney, NSW, 2008, pp. 669-675.
3. P. Liu, G. Huang, Y. Zhou, D. Qin and S. Liu, "Server load based prefetching strategy for P2P VoD streaming," Proceedings of 2013 3rd International Conference on Computer Science and Network Technology, Dalian, 2013, pp. 721-725.
4. E. Divya, R. Sivakoumar and P. Anandhakumar, "Reduction of server load using caching and replication in peer-to-peer network," 2012 International Conference on Recent Trends in Information Technology, Chennai, Tamil Nadu, 2012, pp. 458-462.
5. Z. Chena, K. Xue, P. Hong and H. Lu, "Differentiated Bandwidth Allocation for Reducing Server Load in P2P VOD," 2009 Eighth International Conference on Grid and Cooperative Computing, Lanzhou, Gansu, 2009, pp. 31-36.
6. A. Bestavros, "Speculative data dissemination and service to reduce server load, network traffic and service time for distributed information systems", in Proc. ICDE'96:1996 Int. Conf. Data Eng. , New Orleans, LA, Mar.1996.
7. B. Chandrasekaran "Survey of network traffic models" IEEE Commun. Mag. Mar. 1994.
8. Setia Sonia, Verma Jyoti and Duhan Neelam "A novel approach for semantic web prefetching using semantic information and semantic association",big data analytics, 471-479,2018.
9. H. Hassoun, Fundamentals of Artificial Neural Networks. The MIT Press, 1995.
10. P. Sessini, A. Mahanti, Observations on round-trip times of TCP connections. Society for Computer Simulation, vol. 38 (2006), pp. 347-353.

AUTHORS PROFILE

Sonia Setia is a research scholar in YMCAUST, Faridabad. She completed her M.Tech. in CSE from YMCAUST, Faridabad in 2012 and B.Tech in IT from Kurukshetra University in 2007. She is currently an Assistant Professor with the School of Computer Science And Engineering, Lingayas University, Faridabad, where she has taught and conducted research since August 2017. His current areas of research are Web prediction, data mining, Information Retrieval and natural language processing.

Dr. Jyoti is presently working as Assistant Professor in Department of Computer Engineering, J. C. Bose University of Science and Technology, Faridabad, India. She received her Ph.D in Computer Science Engineering from Maharishi Dayanand University, Rohtak in 2011. She has broad research interests in Data Mining, Information Retrieval. She has published more than 30 papers in refereed journals at national and international level.

Dr. Neelam Duhan is presently working as Assistant Professor in Department of Computer Engineering, J. C. Bose University of Science and Technology, Faridabad, India. She received her Ph.D in Computer Science Engineering from Maharishi Dayanand University, Rohtak in 2011. She has broad research interests in Data Mining, Information Retrieval and Databases. She has published more than 40 papers in reputed conferences and refereed journals.