Continuously updating Cache at each and every node for File Sharing using Cache Accumulation and Resource Sharing Methodology

S Venu Gopal [#1], N Sambasiva Rao [*2]
[# 1] Research Scholar, JNTUH, Hyderabad, TS, India
venusekkeri@gmail.com
[*2] Principal, SRITW for Womens, Warangal, TS, India
[*2] snandam@gmail.com

***Abstract*— in Peer to peer (P2P) file sharing systems contribute a sizable portion of the internet traffic, and this expected to increase in the future. Here every consumer act as autonomous agents. Agents may give referrals or provide services to one another to help find trustworthy services. so, here all services are cached. we describe the wonderful caching technique that allows peers to operate individually based on their local policies. so, using our caching technique provide better file sharing even in the case of peer node discarded from the network. In our proposed work applying CAARS method this deals with updating cache at each and every node level and share the file with bandwidth at last we will show evaluated results using simulation.**

Keyword - Sharing, Peer to Peer, Caching

I. INTRODUCTION

The peer to peer file sharing applications such as Napstart and Gnutella has created a ferment of fresh research activity into peer-to-peer architecture[1-2]. Participation in a peer-to-peer system is dynamic and ad-hoc. The challenge of this system is to figure out a architecture and mechanism for organizing the peers in such a way so that they can cooperate to afford a useful service to the community of end users. For example, in a file sharing application, one challenge is organizing peers into a cooperative, global index so that all content can be efficiently and quickly located by any peer in the system.[3][4][5]

In order to appraise a proposed peer-to-peer system, the characteristics of the nodes that choose to participate in the system must be taken and understood into account. For Example, if some of peers in a file-sharing system have high-latency, low-bandwidth congestion network connections to the internet, the system must be careful to avoid delegating popular or large portions of the distributed.

II. SURVEY

We referred several existing solutions one solutions says that load balanced structured peer-to-peer system but in this problem so many number of clusters we need to maintain. so, here maintenance of each and every cluster is the problem, why because each and every time any node can add and remove into and from the network. another solution says design of proxy cache for peer to peer traffic but here there is a security issue, because creating another proxy cache there may be a theft of data which other peer requesting from and send to another peer group[7]. In the above existing solutions there is a chance of more data loss while sharing[6] from one peer to another peer. So, here by considering the several issues we proposed a solution for above existing problem.

ISSN (Print) : 2319-8613
ISSN (Online) : 0975-4024

III. CONSIDER EXAMPLE NETWORK

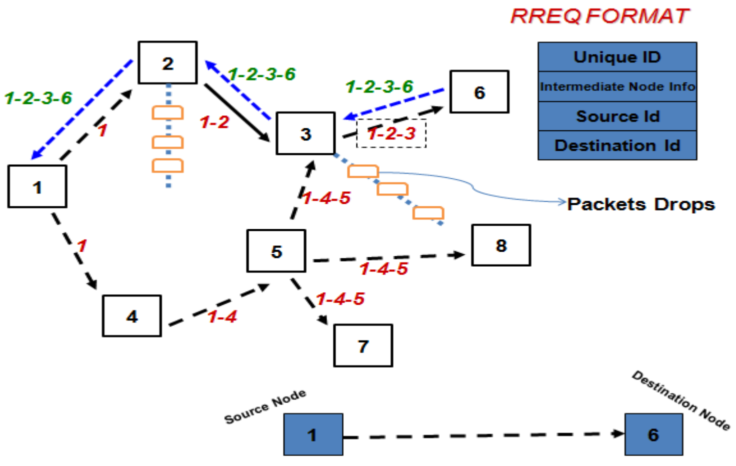


Fig: 01: Example Network

IV. PROPOSED SOLUTION

Here we applied one of the proposed method called P2P-Cache Accumulation and Resource Sharing Method (P2P-CAARS)

Here we are considering the following information for the purpose of simulation and process

it should be CAARS>0

Where B is the allocated bandwidth, PTx denotes the required transmit cache to distribute Here: CAARS Value is not equal to Zero (CAARS=0), so, that we can send and receive data. Simulation time: 10mnts, the achievable data rate could be **calculated as**

$$\text{CAARS} = B\, log_2\left(1 + \frac{P_{Tx}}{N_0}\frac{hr}{B}\right) \text{-----} (1)$$

also calculating load decreased at each and **every node level.**

$$f(node\ p) = \frac{1}{m\ X\ T}\sum_{i=0}^{m} Xi$$

Cache to neighbor peers can only reduce the request messages by one hope. By assuming the weight of request messages is Wr, then per unit time query load decreased by caching to peer **(node P) can be computed as:**

$$Load\ decrease\ (node\ P) = f(node\ p)\ X\ Wr = \frac{Wr}{m\,X\,T}\sum_{i=0}^{m} Xi \text{-----} (2)$$

Peer N Will update the Cache:

1: dest-fileInfo.CaheSource;

2: n sends updating request to dest;

3: if (dest is alive)

4: temp=dest; Equation (1) & (2)

5: else

6: temp=n.find_neighbour(node P)

7: endif

8: do while not temp.Contains(Node P)

9: temp=temo.neughbour (Node P);

10: end do

11: temp sends reply to n;

12: n updates fileInfo from reply message;

V. SIMULATION AND RESULTS

- Number of Nodes : 300
- Source Node : 30
- Destination Node : 287

Table-1:

Packet Size(bytes)	Packets loss (With ache)	Packets loss (Without ache)
64	244	503
128	315	534
256	149	594
512	260	541
1024	557	1550
2048	841	2471
4096	921	4279
8192	3592	7885

ANALYSIS-1

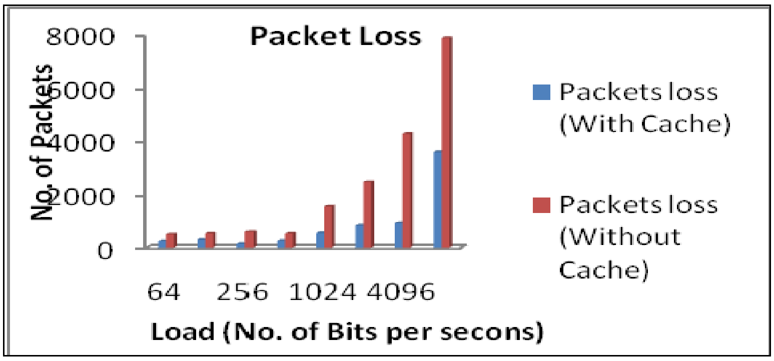


TABLE-2:

Packet Size (bytes)	Packet Delivery Fraction (%) (With Cache)	Packet Delivery Fraction (%) (Without Cache)
64	72.92	44.17
128	65.04	40.73
256	83.46	34.07
512	71.14	39.96
1024	69.09	13.98
2048	68.89	8.58
4096	79.56	5.02
8192	55.70	2.76

ANALYSIS-2:

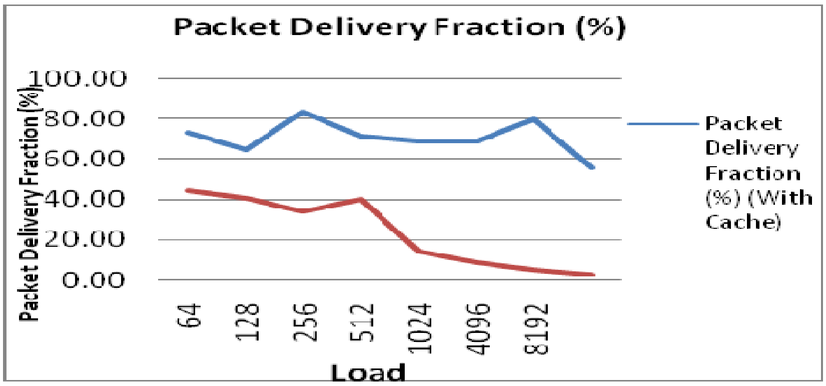


TABLE-3:

Packet Size(bytes)	Throughput (kbps) With Cache	Throughput (kbps) Without Cache
64	51.23	31.64
128	102.45	84.77
256	204.86	123.3
512	332.09	164.22
1024	385.28	188.97
2048	346.79	190.33
4096	376.7	191.66
8192	369.5	193.33

ANALYSIS-3:

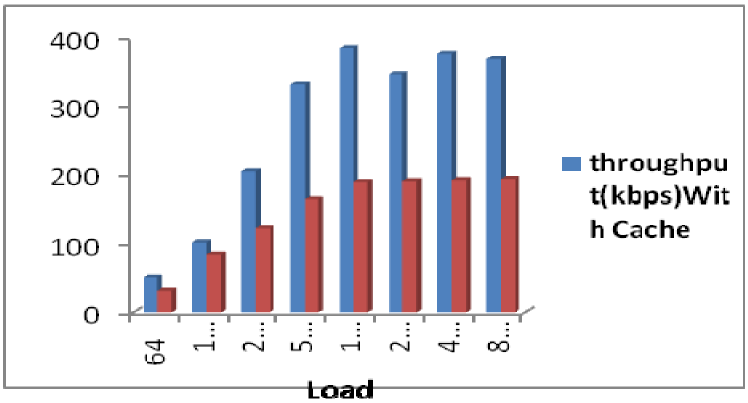
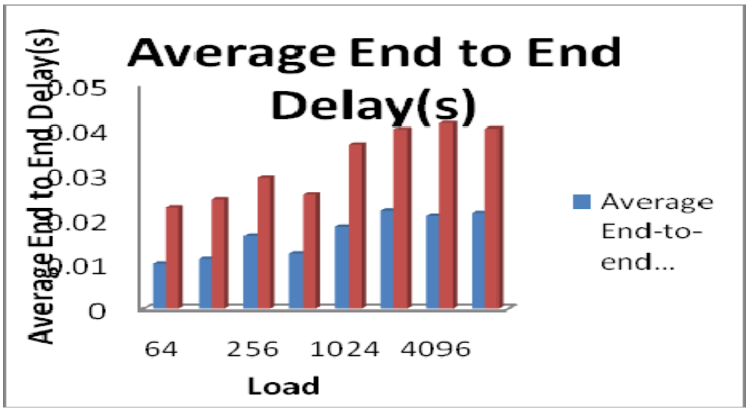


TABLE-4:

Packet Size(bytes)	Average End-to-end delay (With Cache)	Average End-to-end delay (Without Cache)
64	0.009999	0.02261
128	0.011126	0.02452
256	0.016246	0.029316
512	0.012361	0.025415
1024	0.018353	0.03651
2048	0.021891	0.039987
4096	0.020678	0.04141
8192	0.02132	0.040175

ANALYSIS-4:



VI. CONCLUSION

Using our proposed work we can decrease the load at each and every node level so that we can share the data in case of node add or discard from the network, also maximum data we can get from the source to destination. data loss is very low compare to existing solutions.

REFERENCES

[1] I. Clarke, O. Sandberg, B. Wiley, and T. Hong, “Freenet: A distributed anonymous information storage and retrieval system,” in Proceedings of the ICSI Workshop on Design Issues in Anonymity and Unobservability, (Berkeley, CA), 2000.
[2] P. Druschel and A. Rowstron, “Past: A large-scale, persistent peer-to-peer storage utility,” in Proceedings of the Eighth IEEE Workshop on Hot Topics in Operating Systems (HotOS-VIII), (Schoss Elmau, Germany), May 2001.
[3] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, “Chord: A scalable content-addressable network,” in Proceedings of the ACM SIGCOMM 2001 Technical Conference, (San Diego, CA, USA), August 2001.
[4] B. Zhao, K. Kubiatowicz, and A. Joseph, “Tapestry: An infrastructure for fault-resilient wide-area location and routing,” Tech. Rep. UCB//CSD-01-1141, University of California at Berkeley Technical Report, April 2001
[5] S Venu Gopal, N Sambasiva Rao, " Dynamic Sharing of Files from Disconnected Nodes in Peer to Peer Systems," in Proceedings of the IEEE Internation Conference on Electrical, Electrinics, and Optimization Techniques (ICEEOT 2016), Published Nov 2016.
[6] S Venu Gopal, N Sambasiva Rao, "Applying Load Separation Method in Structured Peer to Peer overlay Networks," in Proceedings of the International Journal of Engineering Science and Computing(IJESC) Vol 6 Issue 12, Dec 2016.
[7] S Venu Gopal, N Sambasiva Rao, " Providing File Surveillance to Actively Search for Files by Querying Structured Peer to Peer Overlay Systems," in Proceedings of the An INternational Open Free Access, Peer Reviewed Research Journal Oriental JOurnal of Computer Science & Technology," Vol 8, April 2015

AUTHOR PROFILE

Mr S Venu Gopal, Research Scholar in JNTUH, Hyderabad, Presently working for Vardhaman College of Engineering as Associate Professor in the Department of CSE since 2007. published 3 international journals and one IEEE conference.

Dr N Sambasiva Rao, Principal for SRITW for Womens Warangal, TS, India, with M. Tech & Ph.D and twenty plus years of teaching experience and 6 years of research experience. presently guiding 10 research scholars. he published more than 20 national , International journals and Conferences