Distributed Scheduling algorithm to bringing Fairness to Peer-to-Peer Systems

Kode Durga Prasad, S.Senthil Kumar, D.Saravanan

***Abstract*— Illegal peer-to-peer file-sharing applications suffer from a fundamental problem. Download a lot of free riders, is used to transmit to others by contributing little or no upload bandwidth which slows to download. Contributing as little as possible from the system as much as possible, but taking a lot of free riders, and the lack of quality of service guarantees to support streaming applications. Torrent downloaded from a peer, it is fair, minus the number of bytes indicates the number of bytes uploaded maintains a deficit counter. Torrent free-riders and strategic peers is fair, easy to implement resilient exploit, any bandwidth allocation, the number of peers to estimate rates, no centralized control, and need no parameter tuning. Bit Torrent rule changes a Bit Torrent client running inside the Fair Torrent, and other widely-used Bit Torrent clients compared to its performance against the Planet Lab. Our results contribute to a Fair Torrent peers , two orders of magnitude better fairness, up to five times better download provides up to , and live an average of 60 % -100% in swarms Bit Torrent is a good performance .**

***Index Terms*— Peer-to-Peer, Network, Protocol, Torrent, Performance, Distributed.**

I. INTRODUCTION

The Internet has witnessed a rapid growth in the popularity of various Peer-to-Peer (P2P) applications during recent years. In particular, today's P2P file-sharing applications (e.g., Fast Track, eDonkey, Gnutella) are extremely popular with millions of simultaneous clients and contribute a significant portion of the total Internet traffic . These applications have evolved over the past several years to accommodate growing numbers of participating peers. In these applications, participating peers form an overlay which provides connectivity among the peers, allowing users to search for desired files. Typically, these overlays are *unstructured* where peers select neighbors through a predominantly ad-hoc process this is different from *structured* overlays. Most modern file-sharing networks use a *two-tier* topology where a subset of peers, called ultra peers, form an unstructured sparse graph while other participating peers, called leaf peers, are connected to the top-level overlay through one or multiple ultra peers. More importantly, the overlay topology is continuously reshaped by both user-driven dynamics of peer participation as well as protocol-driven dynamics of neighbor selection. In a nutshell, as participating peers join and leave, they collectively, in a decentralized fashion, form an unstructured and dynamically changing overlay topology.

This work focuses on developing an accurate understanding of the topological properties and dynamics of large-scale unstructured P2P networks, via a case study. Such an understanding is crucial for the development of P2P networks with superior features including better search, availability, reliability and robustness capabilities. For instance, the design and simulation-based evaluation of new search and replication techniques has received much attention in recent year's .These studies often make certain assumptions about topological characteristics of P2P networks (e.g., a power-law degree distribution) and usually ignore the dynamic aspects of overlay topologies. However, little is known today about the topological characteristics of popular P2P file sharing applications, particularly about overlay dynamics. An important factor to note is that properties of unstructured overlay topologies cannot be easily derived from the neighbor selection mechanisms due to implementation heterogeneity and dynamic peer participation. Without a solid understanding of the topological characteristics of file-sharing applications, the actual performance of the proposed search and replication techniques in practice is unknown and cannot be meaningfully simulated. In this case study, we examine one of the most popular file-sharing systems, Gnutella, to cast light on the topological properties of peer-to-peer systems.

II. 2. PROBLEM DEFINITION

A. Existing System:

Previous studies that captured P2P overlay topologies with a crawler either relay on slow crawlers, which inevitably lead to significantly distorted snapshots of the overlay , or capture only a portion of the overlay which is likely to be biased (and non-representative) . These studies do not examine the accuracy of their captured snapshots and only conduct limited analysis of the overlay topology. More importantly, these few studies are outdated (more than three years old), since P2P file sharing applications have significantly increased in size and incorporated several new topological features over the past few years.

Manuscript received January 31, 2014.
Kode Durga Prasad, Final Year MCA
S.Senthil Kumar, Assistant Professor, Sathyabama University, Tamil Nadu, India.
D.Saravanan, Assistant Professor, Sathyabama University, Tamil Nadu, India.

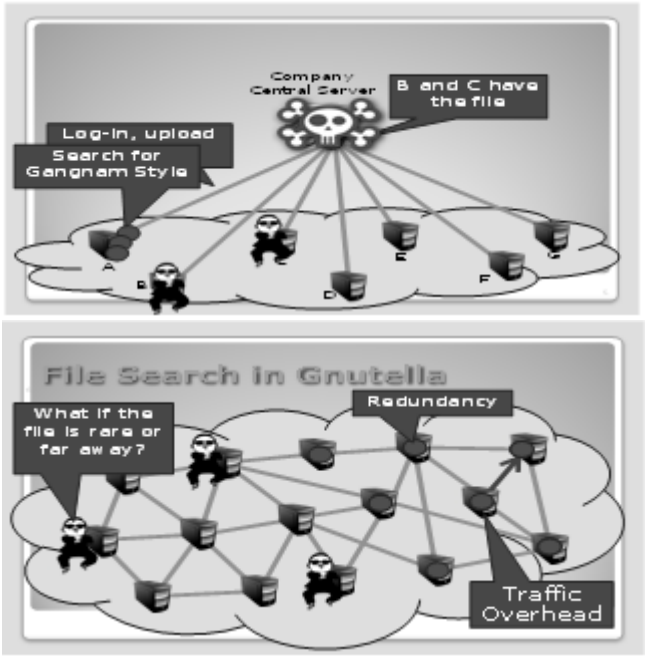

Fig 1. Existing System

B. *Proposed System:*

Precisely capturing the overlay topology of a large scale p2p network is demanding. A common approach is to use a topology crawler that progressively queries peers to determine their neighbors. The captured topology will be a snapshot of the system as a graph; the peers will be represented as vertices and the connections as edges. However, capturing accurate snapshots is inherently difficult for two reasons:

(i) Overlay topologies change as the crawler operates, and

(ii) A non-negligible fraction of peers in each snapshot are not directly reachable by the crawler. When a crawler is slow relative to the rate of overlay change, the resulting snapshot will be significantly destroyed.

Additionally, accuracy verification of a crawler's snapshot is difficult because authoritative reference snapshots are not available. Techniques are introduced for studying the accuracy of a crawler focusing on developing a precise understanding.

III. EXPERIMENTAL SETUP

A. *Fair Torrent Algorithm*

Fair Torrent implements a distributed algorithm that provides fair bandwidth exchange even in the presence of diverse individual peer bandwidth capacities while preserving good download performance. For compatibility with Bit Torrent, Fair Torrent uses the same Bit Torrent protocol, torrent files, and tracker service. Fair Torrent is executed individually by each peer and does not relay on any global allocation or management service beyond what is already provided by Bit Torrent. To describe the Fair Torrent algorithm, we use the definitions of seeds and leechers from Bit Torrent and the terminology in Table 1. Section 3.1 describes the deficit-counter- based main routines of Fair Torrent which exchange data between leechers. Further, describe other important considerations including an even-split seed behavior, a new method for dealing with unchoking, and dynamic considerations.

Procedure 1 : (RECVPACKET) is executed by Li whenever a packet from some peer j is received by Li. RECVPACKET checks that peer j is a leecher. If peer j is a leecher, Fair Torrent increments Recvij and decrements DFij by the number of bytes received from Lj, and re-inserts Lj into the SortedPeerList sorted from lowest to highest deficit values DFij . For simplicity, ties between deficit values are broken using unique peer IDs.

Procedure 1(RECVPACKET)

if $IsLeecher(j)$ and $ValidBlock(p)$ **then**
$Recv_{ij} \leftarrow Recv_{ij} + size(p)$
$DF_{ij} \leftarrow DF_{ij} - size(p)$
$SortedPeerList.ReInsert(j)$
end if

L_i	Leecher i
μ_i	Upload rate of L_i
$Sent_{ij}$	Total bytes sent by L_i to L_j
$Recv_{ij}$	Total bytes received by L_i from L_j
DF_{ij}	Deficit of L_i with respect to L_j: $DF_{ij} = Sent_{ij} - Recv_{ij}$
$Sent_i$	Total bytes sent by L_i to leechers
$Recv_i^L$	Total bytes received by L_i from leechers
$Recv_i^S$	Total bytes received by L_i from seeds
$E(i)$	Instantaneous service error of L_i: $E(i) = Sent_i - Recv_i^L$
$E(i,t)$	$E(i)$ at time t
$E^+_{\max}$ or $E^+_{\max}(i)$	$\max_t E(i,t)$. Max positive service error of L_i
$E^-_{\max}$ or $E^-_{\max}(i)$	$\max_t(-E(i,t))$. Max negative service error of L_i
$E_{\max}$ or $E_{\max}(i)$	$\max(E^+_{\max}(i), E^-_{\max}(i))$ Max service error of L_i
EM	$\max_i(E_{\max}(i))$ Maximum service error.
$packet\_size$	Maximum message size

Table 1. Abbreviations List

Procedure 2: (SENDPACKET) is executed by Li when it is ready to send a packet. Each peer has an upload rate µi, which is expressed in KB per second. Thus, every 1/(µi/packet size) seconds, Li calls procedure SENDPACKET, which tries to pick a leecher with the lowest possible value of DFij . It examines the SortedPeerList starting at the lowest index (which contains the peer with the lowest DFij) and picks the first peer j0 from whom there is a pending request and the connection is writable (i.e. there is room in the TCP socket buffer). Fair Torrent tries to send a packet of up to packet size bytes, but then increments Sentij0 and

DFij0 with the bytes that were actually sent to j0 and reinserts j0 into the SortedPeerList. Fair Torrent uses a packet size of 16 KB for compatibility with older Bit Torrent implementations, and for simplicity given the default 16 KB sub-piece request size in

Bit Torrent. Other Bit Torrent clients typically also use a 16 KB packet size. It is possible that SENDPACKET may not have any data of interest to send to the peer with the lowest deficit. In this case, Fair Torrent just sends data to the next best peer, allowing for maximum utilization of the leecher's upload capacity. Since the deficit DFij with the lowest-deficit peer is always maintained, data will be sent to this peer when it becomes available, and the fairness is preserved. Procedure SENDPACKET assumes the existence of several other procedures. HPRF(j), or HAVEPENDINGREQUESTFROM(j), retur-ns true if there is a pending request from peer j. CWT(j), or CANWRITETO(j), returns true if there is room in j's buffer to send a packet. SEND is the procedure that actually sends the packet from i to j.

Procedure 2 (SENDPACKET)

$n \leftarrow 0; \quad sz \leftarrow Size(SortedPeerList)$
while $(B > 0)$ and $(n < sz)$ **do**
  $j \leftarrow SortedPeerList[n]$
  while !(HPRF(j) and CWT(j)) **do**
    $n \leftarrow n + 1; \quad j \leftarrow SortedPeerList[n]$
  end while
  if $(n < sz)$ **then**
    $use\_bytes = max(B,$ BytesRemainsToSend$(j))$
    $bytes_written \leftarrow$Send(j, use_bytes)
    $B \leftarrow B - bytes\_written$
    if BytesRemainsToSend$(j) == 0$ **then**
      $Sent_{ij} \leftarrow Sent_{ij} + block\_size;$
      $DF_{ij} \leftarrow DF_{ij} + block\_size;$
      $SortedPeerList.ReInsert(j)$
    end if
    $n \leftarrow n + 1;$
  end if
end while

IV. CONCLUSIONS AND FUTURE WORK

Fair Torrent is a fully distributed p2p algorithm which provides each peer with better service proportional to its bandwidth contribution. Fair Torrent's deficit-based distributed algorithm is free from some demerits of previous methods that suffered from slow peer discovery, inaccurate bandwidth estimates, bandwidth under utilization and complex tuning of parameters. Fair Torrent does not require bandwidth estimates, a centralized system, peer reputation, or third-party credit-keeping services. We compared Fair Torrent against Bit Torrent, Azure us, Prop Share, and Bit Torrent. We have demonstrated that because of its high degree of fairness as compared to other p2p systems, Fair Torrent can provide much better performance for participating peers in a number of situations: 30% - 68% better performance in the uniform distribution, 3-5 times improvement for a high uploader in a skewed distribution., 37% - 56% better performance for high contributors in a dynamic situation with line capacities, and 60% - 100% better performance in live swarms. Fair Torrent is resilient to free-riders, low contributors, and strategic peers in both Fair Torrent and Non-Fair Torrent networks. By replacing the high contributors in very popular Azures network, Fair Torrent enhances the performance of not only high contributors but also the entire system, showing that Fair Torrent is adaptable to gradual adoption by users. We are convinced that high fairness and performance guarantees of Fair Torrent establish a strong foundation for developing more reliable and robust p2p services.

Experimental Result:

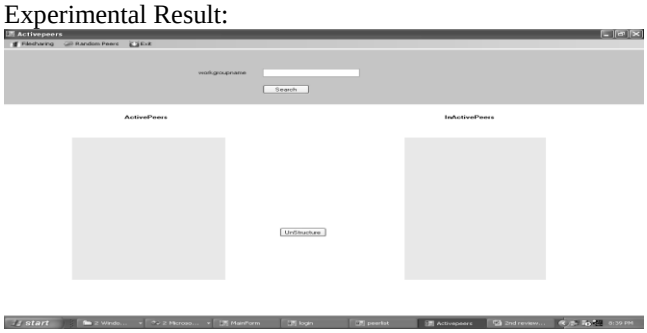

Fig 2.Active peers

Fig 3. Random Peer

Fig 4. Requestion System

Fig 5. Uploading File

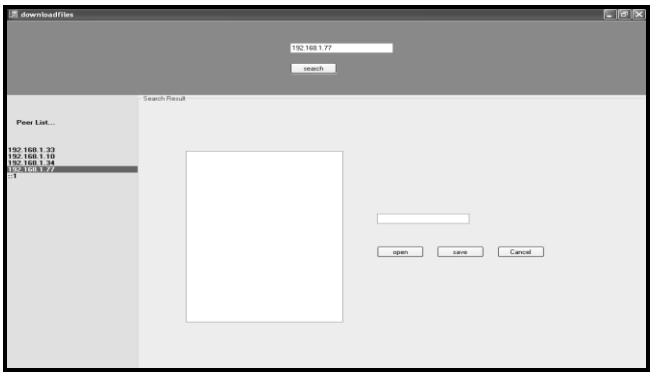

Fig 6. Downloading File

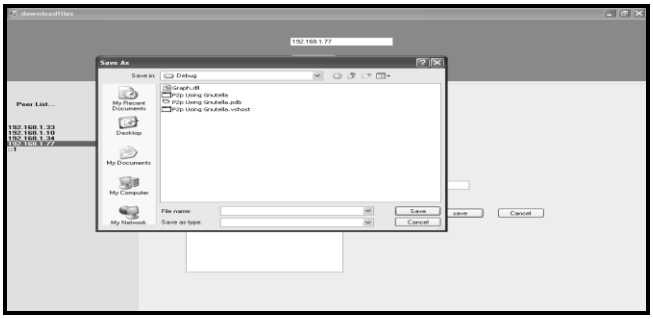

Fig 7. Browse the file

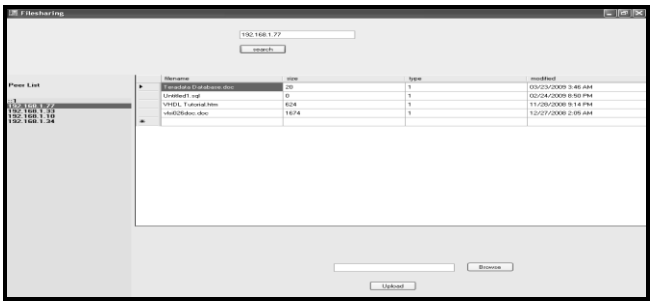

Fig 8. File Sharing

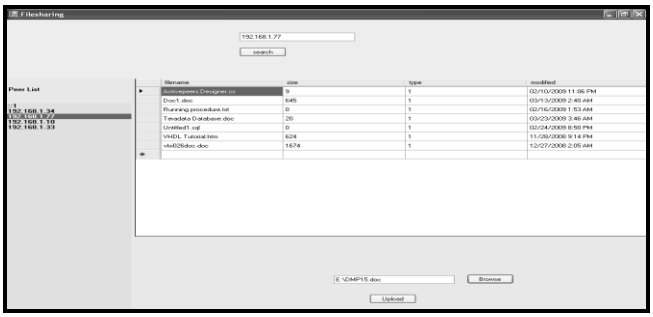

Fig 9. Peer list in file sharing

REFERENCES

[1] A. Aiyer, L. Alvisi, A. Clement, M. Dahlin, J. Martin, and C. Porth, "BAR fault tolerance for cooperative services," in Proc. 20th ACM SOSP, Oct. 2005, pp. 45–58.

[2] D.Saravanan, Dr.S.Srinivasan, "Data Mining Framework for Video Data", In the Proc.of International Conference on Recent Advances in Space Technology Services & Climate Change (RSTS&CC-2010), held at Sathyabama University, Chennai, November 13-15, 2010.Pages 196-198.

[3] Vuze, "Azureus," 2010 [Online]. Available: http://www.azureus.com.

[4] J. C. R. Bennett and H. Zhang, "WF2Q: Worst-case fair weighted fair queuing," in Proc. 15th IEEE INFOCOM, Mar. 1996, pp. 120–128.

[5] D.Saravanan, Dr.S.Srinivasan, "Matrix Based Indexing Technique for Video Data ", International journal of Computer Science", 9 (5): 534-542, 2013,pp 534-542.

[6] K. Berer and Z. Despotovic, "Managing trust in a peer-to-peer infor- mation system," in Proc. ACM CIKM, Nov. 2001, pp. 310–317.

[7] D.Saravanan, Dr.S.Srinivasan, "Video Image Retrieval Using Data Mining Techniques "Journal of Computer Applications, Volume V, Issue No.1. Jan-Mar 2012. Pages39-42. ISSN: 0974-1925.

[8] D.Saravanan, Dr.S.Srinivasan, " A proposed New Algorithm for Hierarchical Clustering suitable for Video Data mining.", International journal of Data Mining and Knowledge Engineering", Volume 3, Number 9, July 2011.Pages 569

[9] A. Bharambe, C. Herley, and V. Padmanabhan, "Analyzing and improving a Bit Torrent networks performance mechanisms," in Proc. 25th IEEE INFOCOM, Apr. 2006, pp. 1–12.

[10] B. Cohen,"Incentives build robustness in Bit Torrent," presented at the 1st Workshop Econ. Peer-to-Peer Syst., Jun. 2003.

[11] F. Cornelli, E. Damiani, S. D. C. di Vimercati, S. Paraboschi, and P. Samarati, "Choosing reputable servents in a P2P network," in Proc. 11th WWW, May 2002, pp. 376–386.