ISSN:0975-9646

Swarm Management in Multi Tracker P2P Network

Vishal Annadate
Computer Science and Engineering, MANIT,
Madhya Pradesh, India.

Abstract—In last couple of years BitTorrent which is one of the unstructured file sharing peer to peer application is being emerged as the most popular paradigm because large number of peers can frequently join and leave the network. In spite of that it doesn't even guarantees that someone is not spreading viruses in swarm. In addition to that exponential and frequent peer participation also badly affect traffic load on single tracker architecture which may result in malfunctioning of tracker or at its worst it will shut down the tracker. This paper proposes multi tracker peer to peer network with swarm management algorithm to pursue efficient response time which is a crucial phenomenon in all kinds without experiencing much performance degradation. In our scenario transitory lifetime of peers is taken into account which trammels tracker to fulfil their query requests within time. Our algorithm ensures that peer gets associated with at least one tracker at a time which ultimately provide better load balancing, stability and response time. Though it seems to be spiders nest but after only a small amount of churn it can be easily incorporated into recent systems and become acclimatize it.

Keywords—Swarm management algorithm, load balancing, multi tracker, transitory lifetime, spiders nest

I. INTRODUCTION

BitTorrent which is purely a peer to peer network have revolutionized the client server architecture and whose distributed nature has become continuously growing archetype and incredible form of internet democracy. It is basically established with nodes having varying capabilities, network bandwidth and storage space but equivalent functionality. They try to utilize their resources and competency to exchange information in much sophisticated manner. Thus, in unstructured centralized peer to peer content sharing systems like BitTorrent [1] nodes connected are tacky to tracker characteristics rather than nodes. This P2P protocol identifies respective nodes through tracker who contributes well with various routing algorithms. Hence in order to evaluate such centralized application trackers characteristic information which is a valuable parameter must be taken in to account. Regardless of tracker configuration peers also plays an important role while dealing with actual download and upload rate.

Though BitTorrent protocols scenario defines totally centralized approach but still it's sufficient enough to handle sufficiently large amount of traffic over network.

Peers in every content distribution cooperative environments has transitory lifetime for query i.e. peers in designated swarm choose to stay connected for finite duration. Due to such limited scope of query in flooding architecture [2] sometimes it's also possible that peers can't locate files even the files available on system though. Hence they are not being able to capitalize on the inherent ability of a system defined.

Actually when peer requests for particular file to tracker, they are responded with random set of peers who are currently downloading or contains complete copy of that desired file. There after peers are going to initiate their query communication within swarm through FTP with peers and HTTP with trackers. As number of nodes joining to the networks grows ultimately size of swarm gets bigger which requires less communication is to be done with tracker and eventually results in comparatively high throughput. To make it happen different download strategies like rarest first, random piece first, endgame mode [1] can be executed to get collection of pieces as metadata from different seed. A seed which is particularly an important component is a peer contain complete copy of desired file should participate in swarm, because except seed its impossible to follow further download.

As far as recent peer to peer communication applications have been concerned, no such policies have been deployed so as to reward seeds with incentives [1] to stay in network. If seed leaves a swarm unexpectedly then peers who are currently communicating with that seed has to wait until he comes in the network if they have not completed their download yet. Intentionally seed never goes down but if happened then peers get stuck for finite interval and after completion of time-stamp they are automatically forwarded to locate other seed if any. If not then peer look forward to collect remaining pieces of file and resumed content will be download later, possibly at last. Swarming or fake block attack which is one of the torrent poisoning method aims straight onto seeder and/or tracker which will cut down swarm and eventually result in either seeder or tracker shut down. Unlike seed if tracker goes down then it doesn't halts current connections or downloads but to accomplish further. So peers in designated swarm are going to stuck for infinite period of time with incomplete downloads or until tracker comes in.

To the best of our knowledge this fact is almost impossible to digest that peers has to wait unconditionally in ideal state. A requirement of recent generation is to exchange huge volume of data with in small amount of span so it's quite risky to handover that total responsibility to single tracker. As tracker is only way of communication in BitTorrent, hence to resolve this condition we must move on to implement some austere network architecture without imbalance of original posture.

The fundamental problem that defies in BitTorrent protocol is if we are using set of trackers then, it's necessary for peers to toggle their tracker dependency in pooled fashion. Therefore here we have proposed new approach which has multiple trackers whose coordination leads to inspect better downloading and uploading rates between them and with peers too. With ease of administration we modified a BitTorrent network with coupled trackers having identical specifications, which make it happen to balance load without assisting each other. A key strength of our approach is removal of single point failure with swarm managing mechanism between trackers which scales networking environment to its higher level. On account of prior work we get some realistic and practical approach towards peer to peer application with negligible of point failure.

In our work we have defined better searching and load balancing procedure of seed without experiencing much unwanted traffic over networking media. Here we've also eliminated splitting and merging headache of swarm with implementation of efficient load balancing algorithm. Our approach states that when peers in cluster requests for a particular query it will be directed towards both the trackers simultaneously. Nevertheless our mechanism specifies both trackers will respond collectively with compared set of peers. This redistribution of peer information is followed with help of algorithm which is going to run on each tracker. In our paper we also emphasized balancing query requests [4] to have better synchronization among peers and tracker than increasing seed instead.

Though it seems to be little bit tricky to handle queries with couple of trackers but they can be easily collaborated with global network. These trackers provide more searching proficiency as higher level interface is may be proxy server or virtual node.

II. Confront Problem

Surprisingly, in recent year's popularity of file sharing application like BitTorrent is at its peak. So it'll ultimately encourages large number of nodes to get attach to swarm and share chunk of content between them. When peers number increases above the level they are expected it will obviously result in increased traffic across swarm that may cause tracker and/or seed failure. As tracker is backbone of BitTorrent so, when it goes down network throughput drops drastically. When tracker becomes unreachable for the peers then ultimately no further communication among peers is going happen.

Potentially, tracker is overloaded with bulk amount of query messages which might not be handled by single tracker. Actually, this serious problem is going to happen in every centralized environment when incoming load limit reached above threshold of tracker. It eventually involves poor load balancing and lack of effective swarm management.

Thus, to resolve this problem we need to make an improvement and have more than one tracker who is capable enough to redirect load among each other without loss of network performance and increase in traffic.

Certainly, it's a trackers responsibility to check status of peer before getting associated. Hence manage such self configured trackers we have to have some concrete base algorithm with which we can reach to completion. In particular we need to face such single point failure problem due to sudden growth of pees in cluster and tracker workload. After recognizing such serious problem that going to occur in file sharing application frequently we must have some reliable and promising solution in order to reduce it.

As per our brief exercise on various peers to peer platforms we observed fluctuations in (downloading and uploading) information gathered at peers due to insufficient use of network bandwidth. BitTorrent protocol currently does not have any such mechanism so as to measure and minimize this kind of conditional delays. We must consider this conditional delay parameter at the time of framework modification because systems overall performance also gets influenced by this factor.

III. Related Work

BitTorrent which is one of the most widely used peer to peer application, lots of work has been done which revolves around its performance measurement and security issues. So, while looking forward it's important to note various security issues like denial of service, swarming, fairness in sharing [3] which may threaten or even collapse our whole network. Perhaps it's worthless to move forward without making any provision to restrict intruders and this might be removed up to some extent with random tracker selection. John Hoffman [5] have proposed multi tracker multi-tier architecture BitTorrent model so that if one tracker fails to respond then query will be forwarded to next tier sequentially. However due to some limitations like poor load balancing, lack of communication leads to traffic overhead.

It's true that that smaller swarm degrades throughput with loss of network bandwidth than large ones for same number of peers. G. Dan and N. Carlsson [4] have explained much efficient way so as to enhance BitTorrent performance using dynamic swarm management (DISM) with couple of trackers. Actually, special tracker redirect message is taken into account so that trackers switching is done efficiently. Added, threshold parameter which defines maximum number of peers in single swarm is considered to split or merge swarm as mentioned in DISM algorithm. This DISM approach is especially applicable on smaller swarms to reduce network traffic at large amount by connecting them to single tracker at a time. But, this algorithm is not going to work when swarm size exceeds over the threshold value. This scenario also gets suffered from inadequacy in load balancing because more than half of peers' requests for same torrent file simultaneously and as such restricts network scalability. After all we can measure network scalability on the basis of work done by seed with number of leechers rather than network size.

Bram Cohen [1] has mentioned importance of providing incentives to encourage more and more peers to be seed in swarm. Nonetheless none of peer to peer application has adopted any special algorithm to give incentives to previously participated seeds of swarm.

Authors Z. Peng, Z. Duan, J.-J. Qi, Y. Cao, and E. Ertao Lv [2] have explained combination of unstructured and structured frameworks which is good for better searching, stability but it's still centralized at lower level. So rather than relying on such centralized archetype our goal must be to have better disparity without disturbing original architecture.

IV. Proposed Algorithm

A. Basic Lookup

As we all know BitTorrent protocols layout is particularly divide into nodes. Hence to make it grow we need to provide some extra backbone for entire network. Here, instead of making much space fragmentation we are dealing with single swarm multi point infrastructure. So far, only few mechanisms have been proposed who work on more than one tracker but this one truly a new angle to look at.

Algorithm includes effective and detailed implementation of query services so as to handle network traffic. Firstly, we are going to introduce some abbreviations, N_P - Network of peers, P_i - ith Peer, t- Torrent file requested by peer, PT_1 - Set of peers connected to tracker T_1, PT_2 - Set of peers connected to tracker T_2, $T_1(t)$- Set of peers replied for torrent t by T_1, $T_2(t)$- Set of peers replied for torrent t by T_2, X- Threshold value of peers connected to Pi

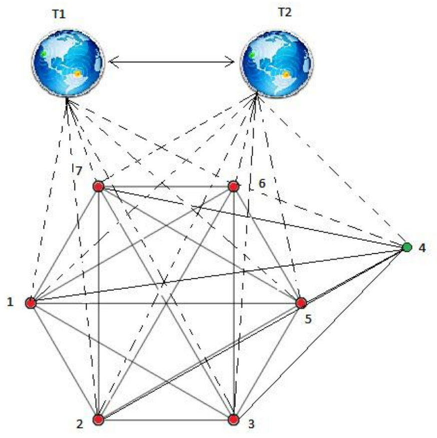

Fig. 1 Multi tracker scenario managing set of peers in cluster

B. Steps to be followed

```
1)  P_i requests for torrent t to both trackers T_1 and T_2 simultaneously.
    $P_i(t) \rightarrow T_1, T_2$ Where $P_i \in N_P$
2)  P_i is acknowledged with T_1(t) and T_2(t) from T_1 and T_2 respectively. $P_i(t) \leftarrow T_1(t), T_2(t)$
    Where, $T_1(t) \leq PT_1, T_2(t) \leq PT_2$
3)  Assume any threshold random value of X (Most probably according to number of peers in net- work).
4)  Condition for swarm would be, if
    $(T_1(t) \geq X \; \&\& \; T_2(t) \geq X)$ {
        If $(T_1(t) < T_2(t))$ $P_i(t) \rightarrow T_1(t)$
        Else
        $P_i(t) \rightarrow T_2(t)$
    }
    Else if $(T_1(t) < X \; \&\& \; T_2(t) < X)$ {
        If $(T_1(t) > T_2(t))$ $P_i(t) \rightarrow T_1(t)$
        Else
        $P_i(t) \rightarrow T_2(t)$
    }
    Else if $(T_1(t) < X < T_2(t))$ {
        If $((X-T_1(t)) < (T_2(t)-X))$ $P_i(t) \rightarrow T_1(t)$
        Else
        $P_i(t) \rightarrow T_2(t)$
    }
    Else {
        If $((T_1(t)-X) < (X-T_2(t)))$ $P_i(t) \rightarrow T_1(t)$
        Else
        $P_i(t) \rightarrow T_2(t)$
    }
5)  End
```

C. Algorithm in Detail

As far BitTorrent is concerned its mandatory to connect each and every peer to tracker however, in our approach it's extended up to get connected to at least one. As we should have to make sure that each peer is getting connected to each other. In our algorithm too query is forwarded to tracker simply as plain flooding but its just that we are going to have probably more number of peers in swarm than that of single tracker framework. And as number of peers joining a swarm grows it will ultimately increases overall throughput.

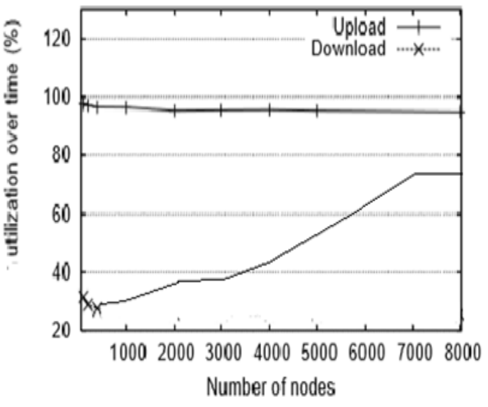

Fig. 2 Bandwidth utilization for upload and download in Multi Tracker Scenario.

As soon as peers request for desired piece of file to download then that message is forwarded to both the trackers simultaneously. As an acknowledgement both the trackers will send back a list of peers in which that piece of file resides and peers who are currently downloading those contents simultaneously. These sets of peers (including seed) which are subset of total number of peers connected to respective tracker are may be different due to attachment flexibility to any of the tracker. So after comparison of both sets we will be having final swarm with unique and useful peers who are connected to any of the tracker depending upon the situation. In this underlying architecture of BitTorrent we can spread huge number of peers randomly to provide much more parallel download.

Here, Fig.2 focuses on utilization of network bandwidth in form of uploading and downloading. So after implementation of this algorithm bandwidth is utilized to its optima for uploading like most of other scenarios. But in case of downloading it seems to be increasing gradually up to much higher level which was not happened in any of such.

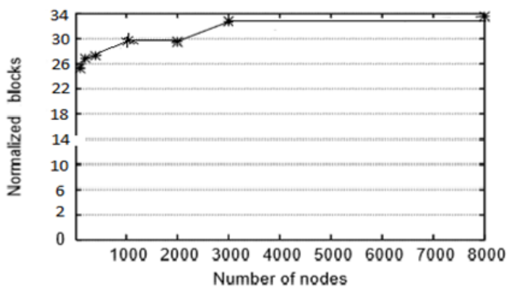

Fig. 3 Seed Contribution in term of number of blocks in Multi Tracker Scenario.

Eventually, however conjecture about increased traffic due to multiple trackers is almost removed with use of swarm management technique. In Fig. 3 blocks exchanged between seed and other peers in multi tracker with swarm management algorithm is tracked. In this case maximum number of blocks transferred by seed is almost double the value than that of single tracker architecture.

Despite of limiting number of client's participation up to some threshold it's important to design much stable and balanced network model to deal with this problem. So, with this algorithms implementation we succeeded to conquer better stability with maximum number of peers in swarm. Apart from tracking peers downloading and uploading status is also been maintained periodically by any of the server.

A tactical advantage of this algorithm is to have random load sharing with which we can scale our network structure to higher level and reduces approximate probability of query failure. Due to provision of multiple trackers at least one of tracker will remain connected even if another tracker goes down.

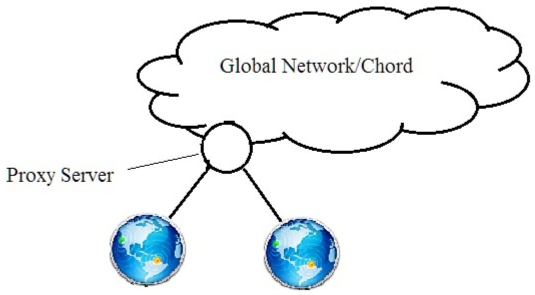

Fig. 4 Communication of trackers with global network through proxy server.

Fig. 4 shows up that when we want to communicate to external global environment then query translation is need to be followed and for that Proxy servers are useful. This implementation will play a crucial role in illuminating performance changes within specific peers which differ significantly as peer participate or leave the network. Actually each peer is responsible for successful communication and maintaining their information in dynamic environment.

V. Conclusion

This swarm management framework with multiple trackers is able to manipulate very large number of peers at a time without experiencing too much response delay or performance degradation. It's probably much effective way as it continues to work for queries that arrive and departures for peers who are partially connected to tracker.

We believe that real deployment of such modified BitTor- rent infrastructure is quite easy and we can configure and further add various new technical issues later with ease.

As far as trackers specification is concerned, both must be identical so that they get synchronized and can be configured in future easily. One more attractive feature of this multi tracker BitTorrent is accuracy, simple to implement and adopt. Moreover it doesn't add too many fluctuations in balancing load as cooperative nature of trackers along with each other's participation for support is sufficient enough to manage it intelligently.

As in recent century fruitful and appropriate response followed with great as well as notable performance is more important than instantaneous response.

References

[1] B. Cohen, "Incentives build robustness in bittorrent," May 2003. http://bitconjurer.org/BitTorrent/bittorrentecon.pdf.
[2] Z. Peng, Z. Duan, J.-J. Qi, Y. Cao, and E. Ertao Lv, "Hp2p: A hybrid hierarchical p2p network," in Digital Society, 2007. ICDS '07. First International Conference on the, pp. 18–18, 2007.
[3] J. Li, "A survey of peer to peer network security issues," 2007. http://www.cse.wustl.edu/～jain/cse571-07/ftp/p2p/index.html.
[4] G. Dán and N. Carlsson, "Dynamic swarm management for improved bittorrent performance," in Proceedings of the 8th International Confer- ence on Peer-to-peer Systems, IPTPS'09, (Berkeley, CA, USA), pp. 10–10, USENIX Association, 2009.
[5] J. Hoffman, "Multi tracker metadata extension," Feb. 2008. http://www. bittorrent.org/beps/bep_0012.html.